Part III Course M10

COMPUTER SIMULATION METHODS IN CHEMISTRY AND PHYSICS

Michaelmas Term 2005

Contents

Introduction

Simulation and Statistical Mechanics

Computer simulations are becoming increasingly popular in science and engineering. Reasons for this include:

- Moore's law: Computers are becoming faster and with more memory
- Simulation techniques are rapidly improving
- Large user-friendly simulation packages are more and more common.

In particular recent developments of *coare-graining techniques*, where some degrees of freedom are "integrated out", leaving a simpler and more tractable representation of the underlying problem, hold much promise for the future.

With all these positive developments also come some potential pitfalls. In particular the proliferation of user-friendly packages encourage the use of simulation as a "black-box". Unfortunately there is often a correlation with how interesting the research problem is, and how dangerous it is to use a simulation technique without knowing what goes on "inside". My hope is that this course will, at the least, help you understand better what happens when you use such a package. Or, even better, that you become able to write your own codes to do fun science.

Statistical mechanics is crucial for the understanding of the computational techniques presented here. Many of the issues that we will discuss are direct applications of the fundamental statistical theory introduced in Part II lectures on the subject and are often very helpful in understanding these sometimes rather abstract concepts. Having followed these lectures is therefore recommended. Most of the essential concepts and methods will be briefly recapitulated before we will apply them in simulation. Two excellent textbooks written with this intimate relation between numerical simulation and statistical mechanics in mind are the book of David Chandler(DC), where the emphasis is more on the statistical mechanics side, and the book of Frenkel and Smit(FS), who give a detailed exposition and justification of the computational methodology. A more introductory text on simulation is the book by Allen and Tildesley(AT).

These lecture notes are made up of two separate sections. The first, on Monte Carlo techniques, was written by Dr. Ard Louis, the second, on Molecular Dynamics techniques, by Prof. Michiel Sprik. They contain more information than you will need to pass the exam. We did this deliberately in the hope that they will be a useful resource for later use, and also because many participants in our course are postgrads who may be using a particular technique for their research and would like to see a bit more material.

Simulation and Computers

Simulation is, of course, also about using computers. The present lectures are not intended to teach computer programming or getting the operating system to run your job. The numerical methods we will discuss are specified in terms of mathematical expressions. However, these methods were designed with the final implementation in computer code in mind. For this

reason we have added appendices outlining the basics algorithms in some detail by means of a kind of "pseudo" code which is defined in a separate section.

Ard Louis, Michaelmas 2005

Key textbooks

On statistical mechanics:

PII *Statistical Mechanics*, Part II Chemistry Course, A. Alavi and J.P. Hansen

DC *Introduction to Modern Statistical Mechanics*, D. Chandler (Oxford University Press).

On Simulation:

FS *Understanding Molecular Simulation, From algorithms to Applications* D. Frenkel and B. Smit, (Academic Press).

AT *Computer Simulation of Liquids*, M. P. Allen and D. J. Tildesley (Clarendon Press).

L *Molecular Modelling, Principles and Applications*, A. R. Leach (Longman).

Part III Course M10

COMPUTER SIMULATION METHODS IN CHEMISTRY AND PHYSICS

Michaelmas Term 2005

SECTION 1: MONTE-CARLO METHODS

Chapter 1

Solving integrals using random numbers

1.1 Introduction

The pressure of the second world war stimulated many important technological breakthroughs in radar, atomic fission, cryptography and rocket flight. A somewhat belated, but no less important, advance was the development of the Monte Carlo (MC) method on computers. Three scientists at the Los Alamos National Laboratory in New Mexico, Nicolas Metropolis, John von Neumann, and Stanislaw Ulam, first used the MC method to study the diffusion of neutrons in fissionable materials. Metropolis coined the word "Monte Carlo" – because of their use of random numbers – later (in 1947), although the idea of using statistical sampling to calculate integrals has been around for much longer. A famous early example is named after the French naturalist Comte de Buffon, who, in 1777, showed how to estimate π by throwing a needle at random onto a set of equally spaced parallel lines. This apparently became something of a 19th century party trick: a number of different investigators tried their hand at "Buffon's needle", cumulating in an attempt by Lazzarini in 1901, who claimed to have obtained a best estimate of $\pi \approx 3.1415929$ – an accuracy of 7 significant digits! – by throwing a needle 3408 times onto a paper sheet[1].

A bewildering array of different MC techniques are now applied to an ever increasing number of problems across science and engineering. In the business world, MC simulations are routinely used to asses risk, setting the value of your insurance premium, or to price complex financial instruments such as derivative securities, determining the value of your stock portfolio.

These lectures will focus on the basic principles behind Monte Carlo, and most applications will be to the calculation of properties of simple atomic and molecular systems.

1.2 Why stochastic methods work better in high dimensions

But first let us investigate a simple variation of Buffon's party trick: If you were so bad at darts that your throws could be considered as truly random, then it's not hard to see that

[1]Lazzarini almost certainly doctored his results. You can easily check this by trying one of the many web-based Java applets that do Buffon's needle. See http://www.sas.upenn.edu/ hongkai/research/mc/mc.html for a nice example

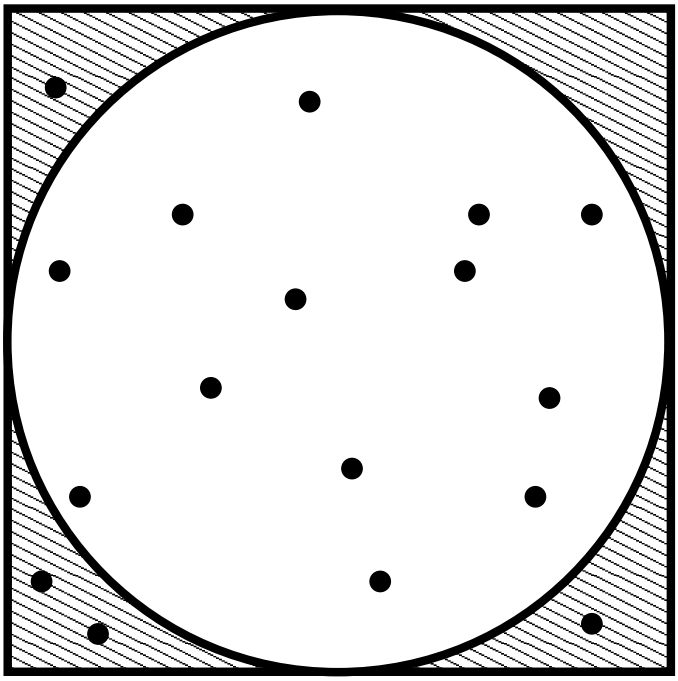


Figure 1.1: π can be calculated by throwing randomly aimed darts at this square, and counting the fraction that lie within the circle. If you do this enough times, this fraction tends toward $\pi/4$. For example, what estimate of π do the random points above give?

the probability of having your dart land inside the circle of Fig. 1.1 would be $\pi/4 \approx 0.785$ of the probability of it landing inside the entire square (just compare the areas). So what you are really doing is evaluating a two-dimensional integral (calculating an area) by a stochastic method.

How accurate would this determination of π be? Clearly if you only throw only a few darts, your value can't be very reliable. For example, if you throw three darts, you could find any of the following ratios: $0, \frac{1}{3}, \frac{2}{3}, 1$. The more darts you throw, the better your estimate should become. But just how quickly would you converge to the correct answer? To work this out, it is instructive to simplify even further, and study the stochastic evaluation of 1-dimensional integrals.

An integral, such as the one described in Fig. 1.2, could be evaluated by selecting N random points x_i on the interval $[0, 1]$:

$$I = \int_0^1 dx f(x) \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i) \tag{1.1}$$

A good measure of the error in this average is the standard deviation σ_I, or its square, the variance, given by $\sigma_I^2 \equiv < (I - < I >)^2 > = < I^2 > - < I >^2$, where the brackets $<>$ denote an average over many different independent MC evaluations of the integral I. Using Eq. (1.1),

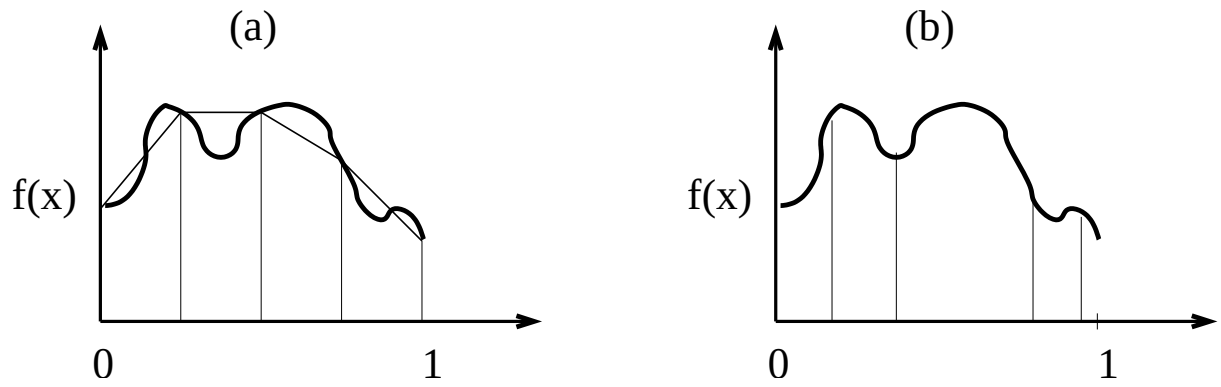


Figure 1.2: This 1-D integral $I = \int_0^1 dx f(x)$ can be calculated in a conventional way (figure (a)), by splitting it up into N segments between 0 and 1, using, e.g. trapezoidal quadrature, or it could be evaluated by MC techniques with random sampling of points (figure (b)). Conventional techniques work best for low dimensions D, whereas MC is better for integrals in high D.

the variance can be rewritten as

$$\begin{aligned}
\sigma_I^2 &= \left\langle \left(\frac{1}{N}\sum_{i=1}^{N} f(x_i) - \left\langle \frac{1}{N}\sum_{i=1}^{N} f(x_i) \right\rangle \right)^2 \right\rangle \\
&= \frac{1}{N^2} \left\langle \left(\sum_{i=1}^{N} (f(x_i) - < f(x) >) \right) \left(\sum_{j=1}^{N} (f(x_j) - < f(x) >) \right) \right\rangle \\
&= \frac{1}{N^2} \left\langle \sum_{i=1}^{N} (f(x_i) - < f(x) >)^2 \right\rangle \qquad (1.2)
\end{aligned}$$

where we've used the fact that the $f(x_i)$ are uncorrelated (which is true if the x_i are uncorrelated), first to write $< I >= 1/N \sum_i^N < f(x) >$ (dropping the index i since x_i is a dummy variable), and then again in last line, where the cross-averages between the i and j sums drop out (i.e.$< (f(x_i) - < f >)(f(x_j) - < f >) >= 0$ if $i \neq j$). Eq. (1.2) can rewritten as:

$$\sigma_I^2 = \frac{1}{N}\sigma_f^2 \qquad (1.3)$$

where σ_f^2 is the (average) variance in $f(x)$ itself, i.e., it measures how much $f(x_i)$ deviates from its average value over the integration region[2]. Since σ_f is, to first order, independent of N, the standard deviation, or average error in the MC evaluation of the integral I of Eq. (1.1), scales as $\sigma_I \sim 1/\sqrt{N}$.

In one dimension (D=1) we can do much better with the same amount of effort by using standard quadrature methods. Even the simple trapezoidal rule:

$$I = \frac{1}{N}\left(\frac{1}{2}f(0) + \sum_{i}^{N-2} f(x_i) + \frac{1}{2}f(1) \right) + \mathcal{O}(\frac{1}{N^3}), \qquad (1.4)$$

[2]Even in this derivation there are one or two subtle assumptions that a purist might snipe at. On the other hand, a much easier way to derive this would be to simply invoke the central limit theorem, from which it follows that the total variance σ_{tot}^2 of N independent statistical samples, each with variance σ_i^2, is given by $\sigma_{tot}^2 \approx 1/N < \sigma_i^2 >$.

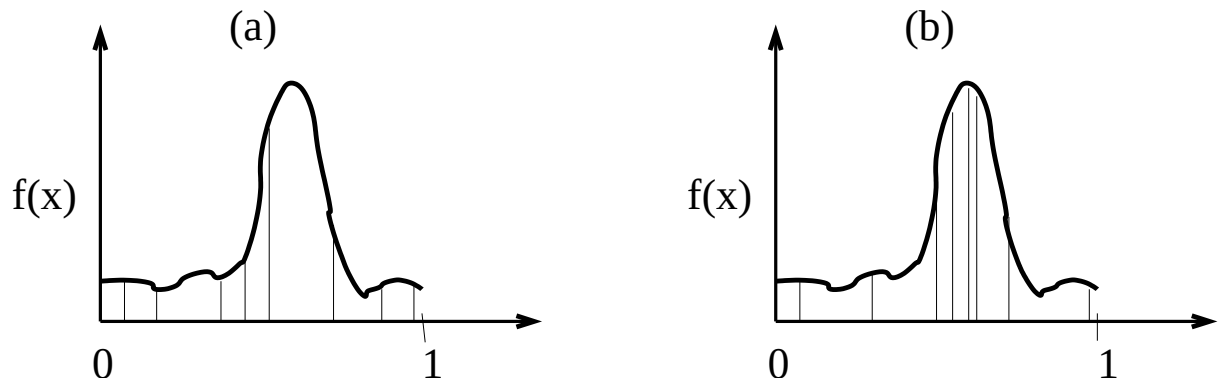


Figure 1.3: In figure (a) the N random points x_i are chosen from a uniform distribution, while figure (b) they come from a biased distribution where points are more likely to occur on a range where $f(x)$ is large. This "importance sampling" can greatly increase the accuracy of a MC integral evaluation.

where the $x_i = i/N$ are equally spaced on $[0, 1]$, scales much better than the MC algorithm. For example, if you quadruple the number of points, the error E_{trap} in trapezoidal rule would go down by a factor $1/4^3 = 64$, while the error in the MC evaluation would merely drop by a factor of 2. In fact, the $1/\sqrt{N}$ scaling implies that if you want to increase the accuracy of your MC calculation by one order of magnitude, you need to do 100 times as much work. Clearly MC isn't the best way to do these 1-D integrals.

The advantages of Monte-Carlo methods only emerge in higher dimensions. This can be seen from simple scaling arguments. Consider an integral over a D dimensional hypercube. Using a standard quadrature method such as the trapezoidal rule, with a fixed spacing of M points per dimension, still gives an error of $E_{trap} \propto M^{-3}$. However now the total number of points is $N = M^D$. The cost of the calculation is proportional to N, which counts the number of independent evaluations of $f(x_i)$ you need to make. Therefore the error in the integral I scales as:

$$E_{trap} \propto M^{-3} = N^{-3/D}. \tag{1.5}$$

In MC, however, the integral's error E_{MC} is independent of dimension (just check the derivation of Eq. (1.2)), and would still scale as

$$E_{MC} \equiv \sigma_I \propto N^{-\frac{1}{2}} \tag{1.6}$$

In other words, MC becomes more efficient than the trapezoidal rule roughly when $N^{-3/D} > N^{-1/2}$, or for dimensions higher than $D \approx 6$. There are more accurate quadrature methods than the trapezoidal rule, but the errors typically scale with the discretisation M, and so for large enough N, MC will always become more efficient. And since many problems in science and engineering require the evaluation of very high dimensional integrals, MC and related stochastic methods are very popular.

1.3 Importance sampling

Sampling points from a uniform distribution, as done in the previous section, may not be the best way to perform a MC calculation. Consider, for example, Fig. 1.3, where most of the

weight of the integral comes from a small range of x where $f(x)$ is large. Sampling more often in this region should greatly increase the accuracy of the MC integration. Of course achieve this, you would need to know something about your function first. But often you do. To make this idea more concrete, let's assume that we are sampling the points from some (positive definite) normalised probability distribution $w(x)$, derived from a best guess for what the function $f(x)$ looks like. The integral in Fig. 1.3 would be rewritten as

$$I = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_{i/w})}{w(x_{i/w})} \tag{1.7}$$

where the w in $x_{i/w}$ is added to emphasise that the x are drawn from the distribution $w(x)$. The division by $w(x_{i/w})$ compensates for the "biasing" of the distribution of the $x_{i/w}$[3]. (Setting $w(x) = 1$ would reduce to uniform sampling.) A very similar analysis to Eq. (1.2) results in the following expression for the variance:

$$\sigma^2_{I/w} \approx \frac{1}{N}\left\langle \frac{1}{N}\sum_{i=1}^{N}\left(\frac{f(x_{i/w})}{w(x_{i/w})} - < \frac{f(x_{i/w})}{w(x_{i/w})} >\right)^2 \right\rangle = \frac{1}{N}\sigma^2_{f/w}. \tag{1.8}$$

Choosing a different sampling distribution $w(x)$ hasn't changed the scaling with N, but it has changed the pre-factor. The best pre-factor would result from using $w(x) = f(x)/ < f >$, in which case $\sigma_{f/w} = 0$! Unfortunately in MC, as in life, there is no such thing as a free lunch: you would need to first know the full integral to obtain $< f >$, which rather defeats the purpose. In practise, however, a good estimate of $w(x)$ may still be available, leading to appreciable gains in accuracy (see the example). Choosing such a distribution is called **importance sampling**, and is a mainstay of almost any MC calculation.

In many cases, especially when sampling highly non-uniform functions, brute force MC evaluations with a uniform distribution of x can result in very large prefactors for the $\sigma_I \propto N^{\frac{1}{2}}$ scaling law[4]. This is exactly the case for the statistical mechanics of atomic and molecular systems, where the high dimensional integrals have a significant contribution in only a very small fraction of the total possible sampling space. Trying to evaluate these integrals without importance sampling would be impossible.

[3]For a more careful derivation see e.g. FS2002, p 25, or try to work it out for yourself by changing variables like you would do for an integration problem.

[4]It is tempting to use the sum in Eq.(1.8), with the $f(x_i)$ drawn from a single run, as an "on the fly" estimator of $\sigma_{f/w}$. This method can be dangerous. Consider the example Fig. 1.3(a) with a limited number of uniform sampling points: $\sigma_{f/w}$ will typically significantly underestimated because the large peak that dominates the integral may not be properly sampled.

Example of 1-d importance sampling

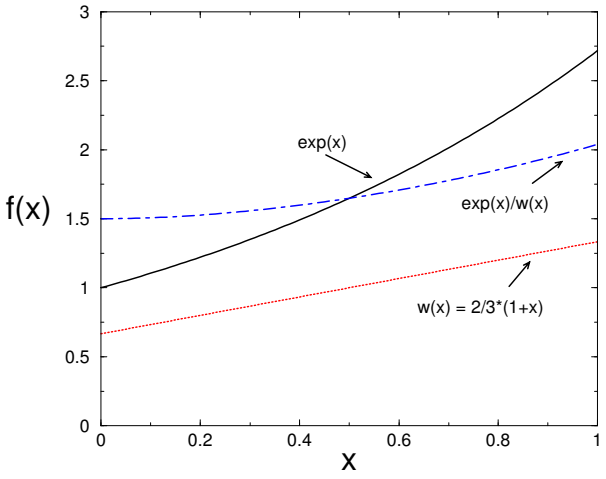
From Eq. (1.2) it follows that the error in an N-step uniform distribution MC evaluation of the integral $I = \int_0^1 dx \exp[x] \approx 1.7182818$ is

$$\sigma_I \approx \frac{0.5}{\sqrt{N}} \tag{1.9}$$

If the normalised importance sampling function $w(x) = 2/3 * (1 + x)$ is used, then the new error, calculated with Eq. (1.8) is:

$$\sigma_{I/w} \approx \frac{0.16}{\sqrt{N}} \tag{1.10}$$

Importance sampling leads to a factor 3 gain in accuracy.



1.4 The Metropolis Monte Carlo Method

1.4.1 Sampling Integrals relevant to statistical mechanics

The summit of statistical mechanics

Statistical mechanics tells us that, given a system at constant number N, volume V and temperature T, the probability p_i of finding it in a microstate i with total energy E_i is proportional to

$$p_i = \frac{\exp\left[-\beta E_i\right]}{Q(N,V,T)} \tag{1.11}$$

where the inverse temperature $\beta = 1/k_B T$, and k_B is Boltzmann's constant. The partition function $Q(N,V,T)$ is defined as the sum over all states:

$$Q(N,V,T) = \sum_i \exp\left[-\beta E_i\right] \tag{1.12}$$

and the average of an operator A is given by:

$$< A >= \sum_i p_i A_i = \frac{1}{Q} \sum_i \exp\left[-\frac{E_i}{k_B T}\right] A_i \tag{1.13}$$

where A_i is the physical value of A for state i. In the words of the great physicist Richard Feynman:

> "*This fundamental law is the summit of statistical mechanics, and the entire subject is either a slide-down from this summit, as the principle is applied to various cases, or the climb-up to where the fundamental law is derived ...*"[5]

We now will begin with slide-down to computing averages with MC techniques.

Why applications of MC to statistical mechanics must use importance sampling

The simplest way to calculate the average in Eq. (1.13) with MC would be to choose M states at random and average:

$$A_M = \frac{\sum_i^M A_i \exp\left[-\beta E_i\right]}{\sum_i^M \exp\left[-\beta E_i\right]}. \tag{1.14}$$

which is analogous to the random uniform sampling for 1-D integrals discussed in the previous section. In theory the limit of an infinite number of sampling points does indeed give $M \to \infty$, $A_M \to < A >$, but in practice there are two major problems:

1. The number of state points in a statistical mechanical system usually grows exponentially with system size.

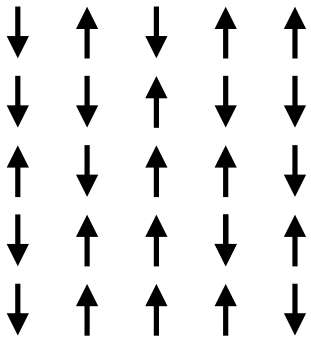
2. Averages like Eq. (1.14) are typically dominated by a small fraction of these states, which random uniform sampling is very unlikely to find.

[5] R.P. Feynman, "Statistical Mechanics", Addison-Wesley, 1972, p1

Consider the following two examples:

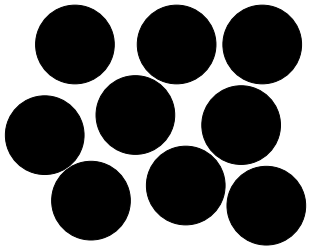
Exponential number of states: spins on a lattice

If you place N spins, constrained to have just two values $S_i = \pm 1$, onto lattice, then the total number of states is proportional to 2^N. Even this simple 5×5 lattice has $2^{25} = 33,554,432$ distinct states. Doubling the length of a side to make a 10×10 lattice, results in over 10^{30} distinct states; the number grows exponentially with lattice size.



Highly peaked distributions: hard spheres near freezing

Hard spheres (HS), round particles that cannot overlap (just like snooker balls), are a popular model for fluids. But even at relatively moderate densities, generating a configuration by assigning random positions to N HS would almost always lead to an overlap which has infinite energy so that the state doesn't contribute to Eq. (1.14). Take, for instance, 100 hard spheres near the freezing transition: only one in about 10^{260} random configurations would count towards the average in Eq. (1.14)[FS2002, p 24]; the distribution is very strongly peaked around a small subset of all possible states.



The simple spin model above demonstrates just how rapidly the number of states can grow with the size of the system. Sampling all states is not an option for anything but the very smallest systems. Luckily, the HS example hints at a way out: A very small subset of all possible configurations dominate the average of Eq. (1.14). Such statistical averages are, in fact, (very) high dimensional analogues of Fig. 1.3, with a few prominent peaks dominating the integral. If we could somehow sample mainly over states in this set of "peaks", we might be able to obtain an accurate average with a reasonable number of MC sampling points. The way forward clearly involves some form of importance sampling as described in section 1.3. Since the Boltzmann distribution (1.11) determines the weight of each state, it is the natural choice of weighting function from which to sample your points. Applying this weighting to the numerator *and* the denominator of Eq. (1.14), and then correcting for the bias as done in Eq. (1.7), cancels the factors p_i to obtain

$$A_M = \frac{\sum_i A_{i/p_i}}{M}. \tag{1.15}$$

The i/p_i reminds us that we sample states over a distribution p_i. This particular importance sampling technique was first described in a seminal paper by Nicolas Metropolis together with the Rosenbluths and the Tellers [M1953]. They calculated the equation of state of hard-discs, and summarised their method in the following words:

> "*So the method we employ is actually a modified Monte Carlo scheme, where, instead of choosing configurations randomly, then weighing them with* $\exp(-E/kT)$*, we choose configurations with a probability* $\exp(-E/kT)$ *and weight them evenly.*"

At first glance this approach doesn't appear to be very practical. First of all, although for many systems the numerator of p_i, $\exp[-\beta E_i]$, is relatively straightforward to calculate, the denominator, given by the partition function $Q = \sum_i \exp[-\beta E_i]$ is almost always impossible to obtain[6]. By analogy again with Fig. 1.3, it's as if we know how to calculate the relative, but not the absolute height of the peaks. Moreover, the space of states i has such a complex high-dimensional structure, that it is very hard to know a-priori exactly where to look for the most probable states, i.e. those with the largest values of $p_i = \exp[-\beta E_i]/Q$, that dominate the averages. However, in their famous 1953 paper, Metropolis *et al*[M1953] devised a very clever way around these problems. Their method, based on a biased random walk through configuration space, is still by far the most popular technique used in atomic and molecular simulations. The next section explains in more detail how this Metropolis Monte Carlo scheme works.

1.4.2 Efficiently sampling phase space with a random walk

Monte Carlo "trajectories"

Sampling configuration space with a "biased random walk" can be described as follows:

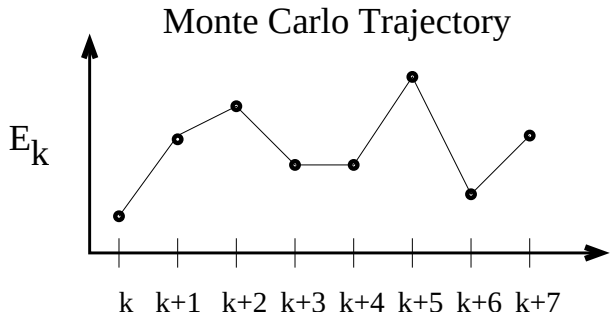
- Start with a given configuration o for which the Boltzmann factor is $\exp[-\beta E_o]$.
- Choose *and* accept a new configuration n, at energy E_n, with a transition probability $\pi(o \to n)$.
- Calculate the value of the operator you are interested in, and add it to your average (1.14)
- repeat to create a MC trajectory through phase space

Note that transition probabilities summed over all other states must add up to unity. This helps define the probability that you stay at the same state o in a MC step:

$$\pi(o \to o) = 1 - \sum_{n \neq o} \pi(o \to n). \tag{1.16}$$

[6] If we could calculate it, we wouldn't need to be doing Monte Carlo!

A Monte Carlo Trajectory:
The Metropolis Monte Carlo algorithm steps through phase space with a random trajectory. If at step k the system is in state o, with energy E_o, then the probability that at step $k+1$ the state will change to n, with energy E_n, is given by $\pi(o \to n)$. The probability that the state at step $k+1$ remains o is given by $\pi(o \to o)$ (see Eq. (1.16)). An example where the state doesn't change would be step $k+3$ to $k+4$. A MC average like Eq. (1.15) should be taken at each step, regardless of whether the state has changed or not.



These stochastic MC trajectories are different from the deterministic trajectories you will encounter in Molecular Dynamics (MD) techniques. They don't need to resemble the realistic dynamics of a physical system at all. In fact, it is exactly this property of "non-realism" that makes the MC technique so useful: one can invent clever methods that sample phase space much more efficiently – nimbly skipping around bottlenecks – than a realistic dynamics would.

Why detailed balance is important

To implement the Metropolis scheme we need to sample configurations from the Boltzmann distribution: the probability of sampling state o should be given by $P(o) = \exp[-\beta E_o]/Q$. How can this be achieved by MC trajectories?

A useful way to think about this is in terms of an ensemble of many MC trajectories, i.e. a huge number[7] of identical physical systems, but with different random walks through the space of all possible states. Then at any given time we can measure $P(o)$ by counting what fraction of walkers are in state o. Once the system has reached equilibrium then $P(o)$ should be *stationary*: the average population of walkers in any state o shouldn't change with time (i.e. MC steps). This implies that the number of systems making a transition to a given state is equal to the number of systems leaving that state. Expressed in mathematical form this statement becomes:

$$P(o)\sum_i \pi(o \to i) = \sum_j P(j)\pi(j \to o). \tag{1.17}$$

(Before you read on, can you see why both $P(j)$ and $\pi(j \to o)$ are included: what is the difference between the two?) In practise though, a more stringent condition is usually imposed:

$$P(o)\pi(o \to n) = P(n)\pi(n \to o). \tag{1.18}$$

which removes the need for sums and satisfies Eq. (1.17). This is often called **detailed balance**: In equilibrium, the average number of accepted moves from a state o to any other state n is exactly cancelled by the number of reverse moves from n to o. Detailed balance guarantees that, once equilibrium is established, the ensemble of random walkers populates the states o with the correct distribution $P(o)$.

[7] It's helpful to think of this number as being much larger than the number of states in a given system

In the Metropolis MC method you need to impose a Boltzmann distribution $P(i) \propto \exp(-E/k_BT)$. Eq. (1.18) suggests that to achieve this, all you need to do is choose the correct transition probabilities $\pi(o \to n)$[8]. It is useful to first split up the determination of $\pi(o \to n)$ into two steps:

1. Choose a new configuration n with a *transition matrix* probability $\alpha(o \to n)$.

2. Accept or reject this new configuration with an *acceptance probability* $acc(o \to n)$.

In other words, the transition probability has been rewritten as:

$$\pi(o \to n) = \alpha(o \to n) acc(o \to n) \tag{1.19}$$

Many MC methods take α to be symmetric, i.e. $\alpha(o \to n) = \alpha(n \to o)$. The detailed balance condition (1.18) therefore implies that:

$$\frac{\pi(o \to n)}{\pi(n \to o)} = \frac{acc(o \to n)}{acc(n \to o)} = \frac{P(n)}{P(o)} = \exp\left[-\beta\left(E_n - E_o\right)\right], \tag{1.20}$$

where only the last equal sign used the fact that the $P(i)$ should follow the Boltzmann distribution. By choosing transition probabilities $\pi(o \to n)$ in this way, which conserves detailed balance, the equilibrium population of MC trajectories will populate the states with the desired Boltzmann distribution. One very clever aspect of this scheme is that there is no need to ever directly evaluate the partition function Q!

1.4.3 The Metropolis Algorithm

Acceptance criteria for the Metropolis Algorithm

There are many possible choices of $acc(o \to n)$ that would satisfy detailed balance and condition 1.20. Here we only discuss the algorithm of Metropolis *et al.*, which, 50 years after its introduction, is still by far the most popular recipe. Their inspired choice was:

$$\begin{aligned} acc(o \to n) &= P(n)/P(o) = \exp[-\beta(E_n - E_o)] \quad \text{if} \quad P(n) < P(o) \\ &= 1 \qquad\qquad\qquad\qquad\qquad\qquad\quad \text{if} \quad P(n) \geq P(o) \end{aligned} \tag{1.21}$$

which means that if the energy decreases you always accept the trial move whereas if the energy increases you accept the move with a probability proportional to the Boltzmann factor of the energy difference. At first sight the two different scenarios for $acc(o \to n)$ may seem strange because they are asymmetric. However, a probability can't be larger than 1, and you should be able to convince yourself that plugging this condition into Eq. (1.20) satisfies detailed balance, and thus leads to the correct Boltzmann distribution of random walkers. (Can you think of other choices for $acc(o \to n)$ that satisfy Eq. (1.20)?)

[8] In principle we could also use Eq. (1.17) to choose the $\pi(o \to n)$ that generate a Boltzmann distribution, but the sums over all states make this more general condition much harder to use than the simpler detailed balance condition (1.18).

Importance sampling: To work out what percentage of yesterday's travellers on the tube slept during their journey, you could sample randomly (or uniformly) over the entire area where you might find them – lets say you're only interested in those who live in England and Wales – or you could do a biased sampling with a much higher probability to test in London, and so gather your data more efficiently. This is the essence of "importance sampling".

Sampling by a drunkards walk in London: But what happens if you don't know the extent of (greater) London? This is analogous to the situation encountered when calculating the (very!) high dimensional integrals needed for statistical mechanics. The method of Metropolis *et al.* solves this problem by using a random (or drunkards) walk algorithm. It works roughly like this: First, start with someone who took the tube yesterday. Ask them **question 1**: *did you sleep on the tube yesterday?* to begin your averaging. Then repeat the following process: Take a big step (or steps) in a random direction. Ask the first person you then see **question 2**: *did you take the tube yesterday?*. If they say yes, ask them question 1, and add the result to your average. Then take new steps in a random direction and repeat. If, on the other hand, they say no to question 2, go back to your original location, and add the original answer to question 1 to your average again. Then take a new random step in a different direction etc...

Here the size and direction of your steps corresponds to the transition matrix $\alpha(o \to n)$, and **question 2** corresponds to the acceptance probability $acc(o \to n)$. In this way you are generating your importance sampling distribution, which is zero for people who didn't take the tube, but finite for those who did. **Question 1**: *did you sleep on the tube yesterday?* is used for averaging, much like is done in Eq. (1.14).

Clearly you need to start somewhere near London to have any chance for your method to work (if you start in the Welsh countryside, you may be stuck there forever). You may also need to adjust the length or number of your steps to optimise your sampling.

The population of people, living in England and Wales, who took the tube yesterday is mainly peaked around London. If you take random samples over all of England and Wales, somewhat like the dots in this picture, your sampling will be very inefficient.

Applying the Metropolis algorithm to a simple fluid

Now that we've finally derived the Metropolis algorithm, let's illustrate it with a more concrete example. Consider a fluid of N particles with positions described by $\mathbf{r}^N = \{\mathbf{r_i}\} = (\mathbf{r_1}, \mathbf{r_2}, ...\mathbf{r_N})$, interacting through a potential $\mathcal{V}(\mathbf{r}^N)$. In the discussion up to now we've always considered the energy E_i of state i, which normally includes both kinetic and the potential energy contributions. But, as shown in the lecture notes for part II Stat Mech **and ...** , the kinetic energy is a rather trivial quantity in classical statistical mechanics[9]. We can easily integrate over the momenta in the partition function to find, as shown in Eq. 1.45 **??** of those notes, that $Q_N = (N!\Lambda^{3N})^{-1} Z_N$, where Λ is the thermal wavelength, and

$$Z_N = \int_V d\mathbf{r}^N \exp\left[-\beta \mathcal{V}\left(\mathbf{r}^N\right)\right] \tag{1.22}$$

is the configurational integral. Since the integral over momentum just gives a constant factor that cancels between the numerator and denominator of Eq. (1.14), we can ignore the momentum and do our MC averages over states that are determined by $\mathbf{r}^N$ alone. In other words, each new set $\mathbf{r}^N$ of fluid particle positions corresponds to a new state i of the system, and the probability distribution $p_i = \exp[-\beta E_i]/Q$ reduces to the following form:

$$P(\mathbf{r}^N) = \frac{1}{Z_N} \exp\left[-\beta \mathcal{V}\left(\mathbf{r}^N\right)\right]. \tag{1.23}$$

In the next box, we summarise the algorithm that Metropolis *et al.*[M1953] introduced for the MC evaluation of thermodynamic averages for a fluid.

[9] as opposed to quantum mechanics where it can be very hard to calculate

Summary of Metropolis Algorithm for fluids

To move from step k to step $k+1$ in the Monte Carlo trajectory repeat the following:

1 select a particle j at random from the configuration of step k:

$$\mathbf{r}_k^N = (\mathbf{r_1}, ..., \mathbf{r_j}, ..\mathbf{r_N}).$$

2 move it to a new position with a random displacement $\mathbf{r}'_j = \mathbf{r}_j + \mathbf{\Delta}$

3 Calculate the potential energy $\mathcal{V}(\mathbf{r}^N_{trial})$ for the new trial state $\mathbf{r}^N_{trial} = (\mathbf{r_1}, ..., \mathbf{r'_j}, ...\mathbf{r_N})$

4 Accept the move $\mathbf{r}_k^N \rightarrow \mathbf{r}^N_{trial}$ with a probability

$$acc(o \rightarrow n) = min\left\{1, \exp\left[-\beta\left(\mathcal{V}(\mathbf{r}^N_{trial}) - \mathcal{V}(\mathbf{r}_k^N)\right)\right]\right\}$$

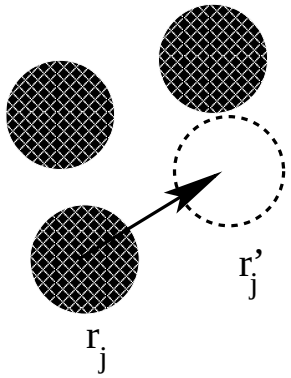
If the trial move is accepted then the state at step $k+1$ is given by

$$\mathbf{r}^N_{k+1} = \mathbf{r}^N_{trial};$$

if it is not accepted then

$$\mathbf{r}^N_{k+1} = \mathbf{r}^N_k.$$

Note that trial moves to lower potential energy are always accepted, whereas moves to a higher potential energy are accepted with a finite probability $\exp\left[-\beta\left(\mathcal{V}(\mathbf{r}^N_{trial}) - \mathcal{V}(\mathbf{r}_k^N)\right)\right]$ that decreases for increasing energy difference.



Trial move of a particle at $\mathbf{r}_j$ to $\mathbf{r}'_j$

5 Add the value of your operator at step $k+1$, i.e. $A(\mathbf{r}^N_{k+1})$, to the average

$$A_{k+1} = \frac{1}{k+1}\sum_{i=1}^{k+1} A(\mathbf{r}_i^N)$$

Go back to step 1 and repeat.

A few comments on the schematic outline above:

- Steps 1 and 2 together define the transition matrix $\alpha(o \rightarrow n)$ of Eq. (1.19) with $o = \mathbf{r}_k^N$ and $n = \mathbf{r}^N_{trial}$. Note that it is symmetric, i.e. $\alpha(o \rightarrow n) = \alpha(n \rightarrow o)$, as required.
- If the random displacement Δ is too large, then most trial steps will be rejected, while if Δ is too small you will only move very slowly through phase-space. The optimum Δ, which leads to the most efficient sampling of phase-space, should be somewhere in between. For many systems a good rule of thumb is to choose an average acceptance of about 50%[10]. For a dense liquid the optimum Δ will be smaller than for a dilute fluid.

[10]This works best for systems with continuous potentials, where the calculation of $\mathcal{V}(\mathbf{r}^N)$ is an expensive step. For systems like hard spheres, a larger Δ, leading to a lower acceptance probability, is often more efficient because you can reject a move as soon as you find the first overlap. A more general way to estimate the optimum Δ for the most efficient sampling of phase-space is to maximise the ratio of the sum of the squares of the accepted displacements divided by the amount of computer time.

(Can you think why this is so?). Since it's hard to tell a-priori what the optimal Δ will be, it is often adjusted once the MC simulation is under way.

- At each MC step only one particle is moved, so that N separate MC steps are roughly equal in cost to a single Molecular Dynamics step, where all particles are moved together according to Newton's equations of motion. You might wonder why we don't also move all N particles together in a single MC step. The reason is quite simple: the main cost in a MC algorithm is usually evaluating $\mathcal{V}(\mathbf{r}^N)$ for a new configuration. Suppose that we can truncate the interactions so that the cost of moving any single particle scales as its average number of neighbours m. The cost of N single moves then scales as mN, which is similar to the cost of a single N particle move. However, if the probability of getting an overlap (and a rejected step) when moving a single particle by a distance Δ is p_{rej}, then the probability of *accepting* a collective move of all N particles scales roughly as $(1 - p_{rej})^N$. To get any acceptances at all p_{rej} would have to scale as $1/N$, which implies an extremely small step Δ. In other words, for roughly the same amount of computational work (i.e. CPU time), a single particle move algorithm advances each particle much further on average than a collective move algorithm does. This is why most MC programs mainly use single-particle moves[11].

1.4.4 Pseudo Code for the MC Algorithm

In this section we use a simplified pseudo-code to describe a MC algorithm for a fluid interacting through a pair potential. (Here I borrow liberally from FS2002)

```
Program MC                                  Basic Metropolis MC Algorithm

  do icycl:= 1, Ncycle                      perform Ncycle MC cycles
       call MCmove                          accept or reject a trial move
       if(mod(icycl,Nsample)=0) then
            call sample                     sample averages every Nsample steps
       endif
  enddo
end
```

description of the subroutines

The subroutine *MCmove*, described in the box below, attempts to displace a random particle The subroutine *sample* adds to averages of the form Eq. (1.14) every *Nsample* steps

[11] Later in the course we describe situations where adding some clever collective moves does lead to a more efficient sampling of phase space.

subroutine MCmove	A routine that attempts to move a particle
*o = int(ran(iseed) * npart) + 1*	select one of *npart* particles at random
call energy(xo,yo,zo,Eo)	energy *Eo* of the old configuration
*xn = x0 + (ran(iseed)-0.5) * delx*	give a particle a random x displacement
*yn = y0 + (ran(iseed)-0.5) * dely*	give a particle a random y displacement
*zn = z0 + (ran(iseed)-0.5) * delz*	give a particle a random z displacement
call energy(xn,yn,zn,En)	energy *En* of the new configuration
if (ran(iseed) < exp(-beta((En-Eo)))* then	acceptance rule 1.21
xo = xn	replace *xo* by *xn*
yo = yn	replace *yo* by *yn*
xo = zn	replace *zo* by *zn*
return	
end	

1.5 Asides on ergodicity and Markov chains

Quite a few assumptions and subtleties were swept under the carpet in the derivation of the Metropolis algorithm described above. Here we'll mention a few of them, some which could have implications for a practical MC algorithm.

- The biased random walk is more formally known as a *Markov process*. This is a general name for stochastic processes without "memory", i.e. the probability to make a step from state o to state n is independent of the previous history of steps.

- A very important assumption is that of *ergodicity*. In brief, this implies that our Markov process can move from any one state i to any other state j in a finite number of steps. Systems with "bottlenecks" in phase-space will often cause problems. Special techniques which mix different kinds of moves can sometimes help overcome this problem.

- The Markov chain should not be periodic. If, for example, you have a two component system, and your algorithm to choose a particle always alternates between species 1 and species 2, then at any given step you always know which species you started from, and you may not sample correctly.

- Under fairly general conditions, it can be proven that a Markov chain approaches the equilibrium distribution exponentially fast. Nevertheless in practise, this can still take quite a few MC steps, and this may also depend very strongly on your starting configuration. When you first start a MC program, you have to *equilibrate* your system – run it until you are satisfied that the correct distribution has been reached – before collecting averages. Equilibration errors are very common[12].

[12] In fact, even Metropolis *et al.* [M1953], probably didn't equilibrate long enough

Example: Equilibration for the Ising Model

The 2-D Ising model, which you saw in Part II practicals, has a Hamiltonian of the following form:

$$H = -\frac{J}{k_B T} \sum_{<i\ j>} S_i S_j + H \sum_i S_i \tag{1.24}$$

where the spins $S_i = \pm 1$, and the double sum is over all distinct pairs. It can be viewed as a very crude model of a magnet. When the external field $H = 0$, then for positive J, the spins can lower their energy by aligning with their nearest neighbours. However, this process competes with entropy, which is maximised when the spins are disordered. For low enough temperature ($k_b T/J \leq 2.29$), this entropy loss is overcome, and the system can spontaneously break its symmetry so that on average the spins will point up or down, behaviour resembling ferromagnetism. A small external field H will then set the direction of the spins, which can be measured by the *magnetisation* M which is defined for an $N \times N$ lattice by

$$M = \frac{1}{N^2} \sum_i S_i. \tag{1.25}$$

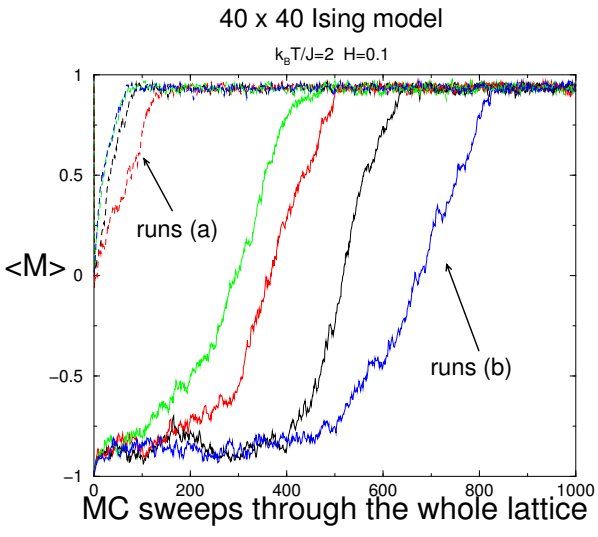
When $H > 0$ then $M > 0$ and we say the spins are pointing up on average, while if $H < 0$ then $M < 0$ and the spins point down on average. We saw earlier how fast the number of states grows with the number of spins; for this reason MC techniques are often employed to study the behaviour of the Ising model[13].

The figure on the right shows the equilibration behaviour of a simple Metropolis MC simulation of a 40×40 Ising model with periodic boundary conditions. Here $k_B T/J = 2$, which is below the transition temperature, and the external field is set to $H = 0.1$, which favours a configuration with most spins pointing up. Two initial conditions were used:

(a) random spin configuration (dashed lines)

(b) all spins down (solid lines)

with 4 independent runs each.



Here are some lessons you can immediately infer from the graph:

- Even for the same initial conditions, equilibration times can vary from run to run.
- Good initial conditions accelerate the approach to equilibrium.
- Averages should only be taken after the system has reached equilibrium.

[13] In 2-dimensions there exists an exact solution by the famous physical chemist Lars Onsager (Phys. Rev. **65**, 117 (1944)), but it is valid only for for $H = 0$. In spite of the simplicity of the model, it has so far resisted all attempts at a solution in 3-D.

1.6 Estimating statistical errors

Steps in MC trajectories are not statistically independent

The variance of an M step MC simulation (measured after equilibrium has been reached), is given by

$$\sigma_M^2 = \frac{1}{M}\sum_{k=1}^{M}\left(A_k - < A >_M\right)^2 = < A^2 >_M - < A >_M^2 \tag{1.26}$$

If the M measurements of A_k were truly independent we could estimate the variance in the average $< A >_M$ by:

$$\sigma^2(< A >_M) \approx \frac{1}{M}\sigma_M^2, \tag{1.27}$$

However, this would be incorrect because the MC trajectories are *correlated*, i.e. it takes a number of steps to move from one independent part of phase-space to another. This rather vague statement can be made a bit more precise by calculating the auto-correlation function:

$$C_{AA}(k) = \frac{1}{M}\sum_{k'}\left(A_{k'} - < A >\right)\left(A_{k'+k} - < A >\right) \tag{1.28}$$

which measures how long it takes for the system keeps a "memory" of what state it was in. This correlation function typically exhibits an exponential decay:

$$C_{AA}(k) \sim \exp(-k/n_\tau) \tag{1.29}$$

and $2n_\tau$ is often taken as the number of steps there are between independent measurements. Therefore, for a MC trajectory of M steps, only

$$n_M = \frac{M}{2n_\tau} \tag{1.30}$$

can be considered to be statistically independent measurements. A better estimate for the variance in the average would be:

$$\sigma^2(< A >_M) = \frac{1}{n_M - 1}\sigma_M^2 = \frac{1}{n_M - 1}\left(< A^2 >_M - < A >_M^2\right) \tag{1.31}$$

which assumes n_M independent measurements[14]. Note that this estimate may be much larger than what you get from naively assuming that all MC steps are independent measurements.

Estimating errors with block averages

Another way to estimate the correct variance, which circumvents the need to calculate a correlation function and the concomitant subtleties in extracting n_τ, is to use *block averages*. In brief the method works like this: Once you have your sequence of M measurements of the

[14]Note that we used the more accurate $1/(n_M - 1)$ factor instead of the more commonly used $1/n_M$ (see any good book on error estimates for how to derive this.). For large n_M the differences are negligible, but this may not always be the case for many realistic MC simulations, where n_M can be rather small.

fluctuating variable A, take L partial averages $< A >_l$ over blocks of length $l = \frac{M}{L}$ Monte Carlo steps each. The variance $\sigma_L^2(A)$ can then be measured as:

$$\sigma_L^2(< A >) = \frac{1}{L}\sum_{i=1}^{L}[A_l - < A >]^2 \tag{1.32}$$

where $< A >$ is the original average over all M steps. As the block size is increased, eventually $L \leq n_M$, so that the measurements become independent, and the laws of statistics predict:

$$\frac{\sigma_L^2(< A >)}{L-1} \approx \sigma^2(< A >). \tag{1.33}$$

$\sigma^2(A)$ is the true variance of average of the fluctuating variable A, measured for N particles in whatever ensemble you used. To achieve this in practise, take your data and plot Eq. (1.33) as a function of L. It should grow with decreasing L, and plateau at roughly the correct variance for $L \leq n_M$[15].

There are quite a few subtleties in calculating errors in MC. For example, even in the same simulation, different properties may have different correlation "times" n_τ. Moreover, there are important differences with how single particle and collective variables behave. See e.g. appendix D of FS2002 for a good discussion of some of these issues.

Systematic errors

Besides statistical errors, which are more straightforward to quantify, a MC simulation can have *systematic errors*. Two of the most common are:

- **finite size errors**

  A MC simulation will, by necessity, always be on a finite size system. If you are interested in thermodynamic averages, these are usually defined for infinitely large systems. The so-called finite size error for many simple properties scales roughly as:

  $$\text{Error} \sim \frac{1}{\sqrt{N}}, \tag{1.34}$$

  where N is the number of particles. Even if you were to do an infinite number of MC steps, you would not converge to the same result as for an infinite size box. In practise, if you are interested in the thermodynamic limit, you can run your simulation for several different numbers of particles, and extrapolate to $N = \infty$, a procedure called *finite size scaling*. However, you do need a good idea of how the property you are interested in scales with system size, which may not always be as simple as Eq. (1.34).

  Besides these statistical errors, finite size simulations may also introduce more serious systematic errors. For example, properties which depend on fluctuations with wavelengths larger than the smallest length of your simulation box will not be sampled properly. The properties of phase-transitions usually show important finite size effects. In general, it is a good idea to test your simulation on several different system sizes until the property you are studying no longer varies.

[15] Once L becomes too small, then the fluctuations in the variance (errors in the error) become relatively large, making it hard to see the plateau. If your simulation was long enough, however, there should still be an intermediate set of L that produces a reliable plateau.

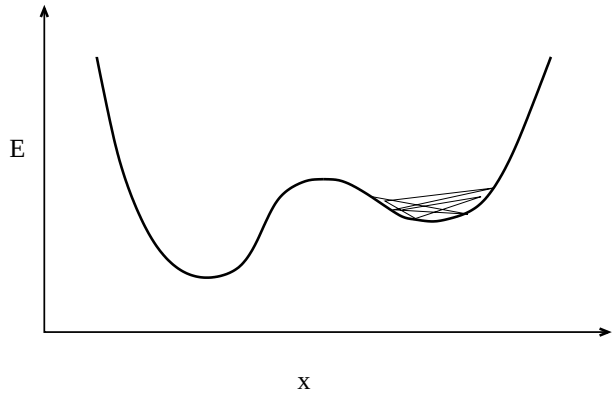


Figure 1.4: Schematic of a simulation stuck in a metastable minimum. This MC simulation trajectory remains in the same energy well, even though it is not the lowest energy state.

- **equilibration errors**

  The simplest equilibration errors occur when averages are taken before the system has reached equilibrium. One obvious way to check this is to monitor the variables you are interested in, and only take averages after they have stopped "drifting". But a more serious problem occurs when the system is stuck in a local minimum of the energy landscape, as Fig. 1.6 shows. Then the system may appear to equilibrate quite nicely inside the local well, even though it is not sampling phase-space correctly (this could also be classed as an ergodicity problem). The runs (b) in the Ising model example show this behaviour: The initial conditions have all spins down while the lowest energy state has all spins up. If you sampled for a short amount of time, (say less than 100 MC sweeps) before the spins collectively flip over, you might erroneously think that the system had equilibrated[16]. As illustrated in that example, where the random configuration equilibrates much faster, it's often a good idea to perform a simulation with several different starting configurations.

[16]In fact, for this particular system, the time to equilibration grows exponentially with decreasing k_BT/J

Chapter 2

Calculating thermodynamic properties with Monte Carlo

2.1 Brief reminder of the Statistical Mechanics of ensembles

The canonical ensemble (fixed N, V, T) is the natural choice for simple MC calculations, just as the microcanonical ensemble (fixed N, V, E) is the natural one for MD simulations. Experiments, on the other hand, are typically performed at constant pressure P. Moreover, considerations of computational efficiency may impose the constraint of other parameters. For example, fixing the chemical potential μ is often useful when studying adsorption processes.

The inherent flexibility of MC moves makes this technique uniquely suited for sampling other ensembles. But before describing details of implementation, we will engage in a "climb up", à la Feynman, to a few related summits, and briefly discuss some of the more popular ensembles, summarised in the table below [1].

Summary of 4 popular ensembles

Ensemble	Thermodynamic potential	Partition function	Probability distribution
microcanonical fixed (NVE)	$S/k_B = \log[\Omega(N,V,E)]$ entropy	$\Omega = \sum_i \delta_{E,E_i}$	$p_i = \frac{1}{\Omega}$
canonical fixed (NVT)	$-\beta A = \log[Q(N,V,T)]$ Helmholtz free-energy	$Q = \sum_i \exp[-\beta E_i]$	$p_i = \frac{\exp[-\beta E_i]}{Q}$
isobaric fixed (NPT)	$-\beta G = \log[\Delta(N,P,T)]$ Gibbs free energy	$\Delta = \sum_i \exp[-\beta(E_i + PV_i)]$	$p_i = \frac{\exp[-\beta(E_i + PV_i)]}{\Delta}$
grand canonical fixed (μVT)	$\beta PV = \log[\Xi(\mu,V,T)]$ Grand potential	$\Xi = \sum_i \exp[-\beta(E_i - \mu N_i)]$	$p_i = \frac{\exp[-\beta(E_i - \mu N_i)]}{\Xi}$

[1] Later sections will hopefully convince you that computer simulations are more than just a slide down towards applications – they often bring into focus the subtle differences between ensembles.

microcanonical

The microcanonical ensemble is in a sense the most basic one. Three extensive variables, the total energy E, particle number N, and volume V are fixed. All states are equally likely, i.e. $p_i = \frac{1}{\Omega}$, so that the partition function $\Omega(E, V, T)$ is a simple sum of the total number of microstates of the system at those fixed parameters. The link between the partition function and thermodynamics proceeds through the entropy[2] $S/k_B = \log[\Omega(E, V, T)]$ from which the thermodynamic definition of the temperature

$$\frac{1}{T} = \left(\frac{\partial S}{\partial E}\right)_{V,N} \tag{2.1}$$

and other thermodynamic properties can be derived.

However, this ensemble is not well suited to Monte-Carlo calculations, because each state is equally probable and the advantages of importance sampling over a small subset of states is lost.

canonical

The canonical ensemble can be derived by imagining a microcanonical system, split into a smaller (but still macroscopic) subsystem I with energy E_i, V_i, N_i, and a larger subsystem II with $E - E_i, V - V_i, N - N_i$, as depicted in Fig. 2.1. The borders between I and II keep V_i and N_i fixed, but allow energy to be shared between the two subsystems. By taking the limit where the size of system II relative to system I goes to infinity – ($(N - N_i)/N_i \to \infty$ with $\rho = N_i/V_i = (N - N_i)/(V - V_i)$ fixed – but still keeping the smaller system macroscopic), system II effectively becomes microcanonical again. The probability of finding the smaller subsystem with an energy E_i is equal to finding the larger system with energy $E - E_i$. The latter can now be viewed in the microcanonical ensemble, so the probability is given by:

$$p_i = \frac{\Omega(E - E_i)}{\sum_j \Omega(E - E_j)} \tag{2.2}$$

where the $\Omega(E - E_i)$ are the microcanoncial partition functions of subsystem II. Since we are treating the limit where system I is much smaller than system II, it makes sense to expand $\log[\Omega(E - E_i)]$ (which, in contrast to $\Omega(E)$, is extensive), in the small parameter $x = E_i/E$ around $x = 0$ which gives:

$$\begin{aligned} \log\left[\Omega(E - E_i)\right] &= \log\left[\Omega(E)\right] + x\left(\frac{\partial \log\left[\Omega(E(1-x))\right]}{\partial x}\right)_{x=0} + \mathcal{O}(x^2) \\ &\approx \log\left[\Omega(E)\right] - \frac{E_i}{k_B T} \end{aligned} \tag{2.3}$$

where in the last line we've changed variables from x to E_i, and used the relationship $S/k_B = \log[\Omega(E, V, N)]$, and the definition of temperature, Eq. (2.1), to simplify. Using Eq. (2.3) in Eq. (2.2) then gives

$$p_i = \frac{\Omega(E)\exp\left[-\beta E_i\right]}{\sum_j \Omega(E)\exp\left[-\beta E_j\right]} = \frac{\exp\left[-\beta E_i\right]}{\sum_j \exp\left[-\beta E_j\right]}. \tag{2.4}$$

[2]This relationship between entropy and the number of states in a system, which Ludwig Boltzmann had inscribed on his tombstone, could also make a strong claim to being the summit of statistical mechanics.

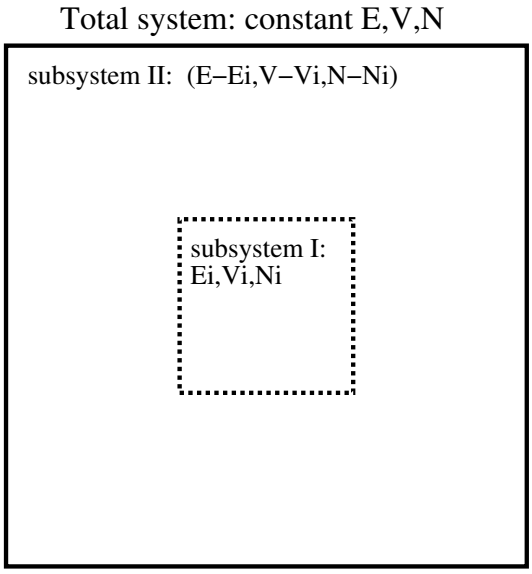


Figure 2.1: Defining a subspace of a larger microcanonical system helps in deriving different ensembles.

This little "climb up to the summit" helps explain why in the canonical ensemble the p_i decrease with increasing E_i: system I is coupled to a much larger system for which the number of states peaks at $x = 0$[3].

Since the energy is allowed to fluctuate, the subsystem takes on the temperature T of the total system, which can be viewed as a heat bath. Thus the relevant parameters for the canonical partition function $Q(N, V, T)$, defined by the denominator in Eq. (2.4), are two extensive variables, N and V, and the intensive variable T. The connection to thermodynamics proceeds via the Helmholtz free energy $\beta A = -\log[Q(N, V, T)]$

isobaric

The isobaric ensemble can be derived in a similar fashion to the canonical ensemble. Now the subsystem I of Fig. 2.1 allows not only energy, but also volume to fluctuate[4]. This means that there is one fixed extensive variable, N, and two intensive variables, the temperature T and the pressure P. The partition function $\Delta(N, P, T)$ and phase-space probabilities are given in the table, while the connection to thermodynamics proceeds via the Gibbs free energy $\beta G = -\log[\Delta(N, P, T)]$. Note that the sum over states includes not only different energies, but also different volumes.

grand canonical

To derive this ensemble simply define an imaginary boundary for system I. The thermodynamic states are determined by the (fluctuating) subset of particles within the prescribed volume. Particles, and with them energy, freely move across the imaginary borders of the box, leading to

[3]see e.g. FS 2002, p 9-13, or a number of standard texts on statistical mechanics for a more detailed derivation.

[4]It's easier to derive this ensemble by first following the subsystem procedure to define a canonical ensemble, and then repeating it with walls that allow volume fluctuations (like a big piston) to derive the isobaric ensemble

two constant intensive variables, the temperature T and the chemical potential μ. The required extensive variable is the volume V[5]. This exactly describes the grand-canonical ensemble. Expressions for the partition function $\Xi(\mu, V, T)$, the probability p_i, and the connection to thermodynamics through $\beta PV = \log[\Xi(\mu, V, T)]$ are given in the table.

averages and fluctuations

The number of particles in a canonical ensemble is fixed at N, whereas in a grand-canonical ensemble N differs from state to state, with relative fluctuations of order $1/\sqrt{N}$. The converse holds for the chemical potential μ. However, for the same system at the same state point, the ensemble averages of each quantity will be equal, at least in the thermodynamic limit. For example, if for a given N, V, T you measure $\mu = \langle\mu\rangle_{N,V,T}$ in the canonical ensemble, then in the grand canonical ensemble, for the same μ, you will find $\langle N\rangle_{\mu VT} = N$[6].

It is important to remember that this equivalence of averages does not hold for *fluctuations*. For some variables this is obvious; consider for example fluctuations in the energy E. Whereas these will be zero (by definition) in the microcanonical ensemble, they will be finite in the other three ensembles discussed above. Fluctuations can often be related to thermodynamic properties, and are therefore useful quantities in computer simulations. But care must always be taken in their interpretation. For example, in the canonical ensemble, the fluctuations in the total energy can be directly related to the specific heat $C_V = (\partial E/\partial V)_{NT}$ at constant volume:

$$\left\langle (E - < E >)^2 \right\rangle_{NVT} = k_B T^2 C_V. \tag{2.5}$$

But this does not hold for energy fluctuations at constant NPT (the isobaric ensemble). Nevertheless, for such systems, the fluctuations in the instantaneous enthalpy $H = E + PV$ can be used to calculate the specific heat at constant pressure:

$$\left\langle (H - < H >)^2 \right\rangle_{N,P,T} = k_B T^2 C_P \tag{2.6}$$

A whole host of other useful fluctuation relations exist, and can be found in standard texts[7]. The take home message is that one should always be careful to use the appropriate fluctuation relationship for the ensemble one is calculating with.

[5]Can you think of reasons why defining an ensemble with three intensive variables causes difficulties for simulations?

[6]Note that this is strictly true only in the thermodynamic limit. In practise a simulation is always for a finite sized system, and the finite size effects may not be exactly the same in each ensemble.

[7]Such as FS2002, or Allen and Tildesley 1987

2.2 Constant pressure MC

Experiments are often naturally performed at constant pressure, so simulations in this ensemble can be quite useful. How does one go about implementing the isobaric ensemble in a MC computer simulation? To answer that, we first specialise to a classical fluid, where, just as for the canonical ensemble, as described in in Eqs. (1.22) and (1.23), the momentum coordinates can be intergrated out of the partition function Δ defined in the table. Averages then take the form:

$$\langle A \rangle_{NPT} = \frac{\int_0^\infty dV \exp\left[-\beta PV\right] V^N \int d\mathbf{s}^N A(\mathbf{s}^N; V) \exp\left[-\beta \mathcal{V}(\mathbf{s}^N; V)\right]}{Z(NPT)} \tag{2.7}$$

where $Z(NPT)$ is generalisation of the configurational integral $Z_N = Z(N, V, T)$ defined in Eq. (1.22) for the canonical ensemble. It is the function that would normalise the distribution. Here we've scaled the particles in the configuration $\mathbf{r}^N = (\mathbf{r}_1, \mathbf{r}_2, ...\mathbf{r}_N)$ by the volume of the box – each $\mathbf{s}_j$ in $\mathbf{s}^N$ is given by $V^{-\frac{1}{3}}\mathbf{r}_j$ – to make explicit the dependence of $d\mathbf{r}^N$ on the volume (This is the origin of the V^N term in Eq. (2.7)). In analogy to Eq. (1.23), valid for the canonical ensemble, a Monte-Carlo algorithm for the NPT ensemble should sample states with a probability distribution proportional to:

$$P_{NPT}(\mathbf{s}^N, V) \propto \exp\left[-\beta\left(\mathcal{V}(\mathbf{s}_N) + PV - N\beta^{-1}\log\left[V\right]\right)\right] \tag{2.8}$$

But now states are not only defined by the configuration $\mathbf{s}^N$, but also by the volume V, which can fluctuate. To properly sample this distribution we need two kinds of MC moves:

1. moves that randomly displace a particle

2. moves that randomly change the volume V

Moves of type 1 were described in chapter 1. Moves of type 2 can also be performed by an adaptation of the Metropolis prescription: The transition probability $\pi(o \to n) = \alpha(o \to n)acc(o \to n)$ is again split into a symmetric transition probability $\alpha(o \to n)$ (the probability to choose a certain trial volume move from V_o to $V_n = V_o + \Delta V$) and an acceptance probability $acc(o \to n)$ chosen such that the MC trajectory reproduces the distribution $P_{NPT}(\mathbf{s}^N, V)$ of Eq. (2.8):

$$acc(o \to n) = min\left\{1, \exp\left[-\beta\left(\mathcal{V}(\mathbf{s}^N; V_n) - \mathcal{V}(\mathbf{s}^N; V_o) + P(\Delta V) - N\beta^{-1}\log\left[\frac{V_o + \Delta V}{V_o}\right]\right)\right]\right\}. \tag{2.9}$$

Even though $\mathbf{s}^N$ itself remains constant, the change of volume scales all the coordinates, and therefore affects the potential energy $\mathcal{V}(\mathbf{r}^N; V)$[8].

[8]It is instructive to examine the case of an ideal gas, where $\mathcal{V}(\mathbf{r}^N) = 0$: the $\log\left[\frac{V_o+\Delta V}{V_o}\right]$ and the $P(\Delta V)$ terms work in opposite directions. Without the log term, any move which shrunk the volume would be accepted, leading to a collapse of the system.

volume moves

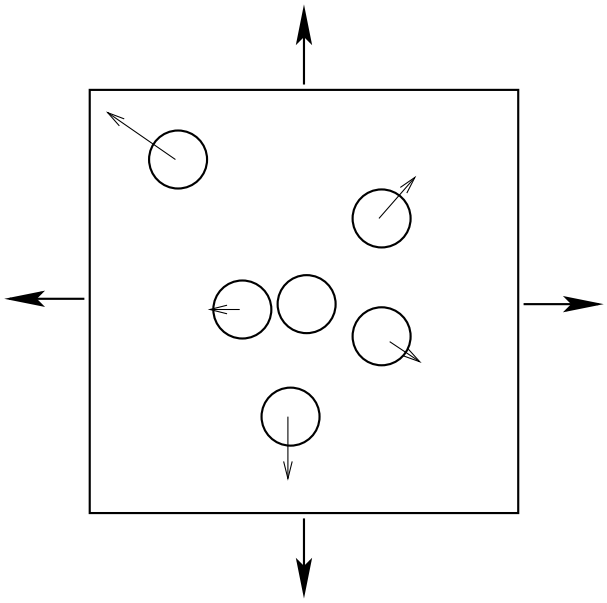
The cost of a volume move depends very much on the type of potentials used. For interactions of the type:

$$\mathcal{V}(\mathbf{r}^N) = \sum_{i<j} \epsilon \left(\frac{\sigma_{ij}}{r_{ij}} \right)^m \qquad (2.10)$$

the effect of a volume change is very simple:

$$\mathcal{V}(\mathbf{s}^N, V_n) = \left(\frac{V_o}{V_n} \right)^{m/D} \mathcal{V}(\mathbf{s}^N, V_o), \quad (2.11)$$

where D is the system dimension. For these potentials, or those made up of sums of such terms (like the Lennard Jones form), a volume move is inexpensive. For other types of interactions a volume move can mean recalculating $\mathcal{V}(\mathbf{r}^{\mathcal{N}})$ for all the new distances, which is roughly as expensive as N separate one-particle MC moves.



Upon a volume move from V_o to V_n (here $V_n > V_o$), in 3 dimensions, all molecule centre of mass distances from the origin are scaled by a factor $(V_n/V_o)^{\frac{1}{3}}$. Note that particles near the edges move further than particles near the origin.

The cost of a volume move depends on the type of potential; for potentials with one length-scale it will be cheap (see box above), but for more complicated potentials, such as those often used in molecular simulations, the move will be expensive. In the former case, volume moves can be attempted as often as particle moves, while in the latter case, they should be attempted much less often, perhaps once every N particle moves. Either way, it is important make the choice of when to take a volume move randomly, and not after a fixed number of particle moves. Otherwise we could violate detailed balance, which states that for any move $o \to n$ we should also be able to make the move $n \to o$. Just as for particle moves, one should optimise the average size of a volume move. Choose it too small and you will find many acceptances, but creep through phase-space very slowly; choose it too large, and the number of accepted moves will be extremely low, which is also computationally inefficient.

```
program MCnpt                                  isobaric ensemble Monte Carlo program

    do mcycl := 1,ncycl                        do ncycl moves
        i := int((natom + 1)* ran(iseed))+1
        if (i ≤ natom) then                    chooses volume move only once in natom tries
        MCmove                                 perform a particle move
        else
        VolMCmove                              perform a volume move
        endif
    enddo
    end
```

2.3 Grand Canonical MC

In the grand-canonical ensemble, particles can be exchanged with a "particle bath", which fixes the chemical potential μ. For a classical fluid, the probability distribution is proportional to

$$P(\mathbf{s}^N; N) \propto \frac{\exp\left[\beta\mu N\right] V^N}{\Lambda^{3N} N!} \exp\left[-\beta \mathcal{V}(\mathbf{s}^N)\right] \tag{2.12}$$

and a state is defined not only by the particle configurations, but also by the number of particles N. Besides the usual particle displacement moves, we now also need particle insertion or removal moves. Again the transition probability $\pi(o \to n)$ can be split up into a transition matrix, $\alpha(o \to n)$ and an acceptance probability $acc(o \to n)$. Besides the usual particle moves, there are now two additional trial moves: 1) removing a randomly chosen particle and 2) inserting a particle at a random position. It is convenient to simply set the probability of attempting a trial insertion or removal to be equal, i.e. to make the transition matrix symmetric: $\alpha(N \to N+1) = \alpha(N+1 \to N)$. In that case we only need to determine the acceptance probabilities, given by:

1. The removal of a particle, chosen at random

$$acc(N \to N-1) = min\left\{1, \frac{\Lambda^3 N}{V} \exp\left[-\beta\left(\mu + \mathcal{V}(\mathbf{s}^{N-1}) - \mathcal{V}(\mathbf{s}^N)\right)\right]\right\} \tag{2.13}$$

2. The insertion of a particle at a random position $\mathbf{s}_{N+1}$, i.e. changing the configuration from $\mathbf{s}^N$ to $\mathbf{s}^{N+1}$

$$acc(N \to N+1) = min\left\{1, \frac{V}{\Lambda^3 (N+1)} \exp\left[-\beta\left(-\mu + \mathcal{V}(\mathbf{s}^{N+1}) - \mathcal{V}(\mathbf{s}^N)\right)\right]\right\} \tag{2.14}$$

The proof that this whole scheme satisfies detailed balance, as well as an example of a pseudo-code, are left as an in-class exercise.

In-class exercises: proof of detailed balance for Grand Canonical MC algorithm

In-class exercise: pseudo-code for Grand-Canonical MC

Example: Calculating the equation of state

The equation of state, i.e. the variation of the pressure P with density ρ and temperature T is a much studied property of liquids. To make the discussion of different ensembles a bit more concrete, we briefly describe how to calculate $P(\rho)$ along an isotherm (constant T), for each of the following three ensembles:

canonical ensemble Here N, V, T are fixed, and so a typical procedure would be to fix the number of particles N and temperature T, and calculate the pressure $< P >$ at a number of different volumes V, using the virial equation described in the notes of Dr. Sprik.

isobaric ensemble Here N, P, T are fixed, and so one simply fixes N, P, and T, and lets the system find its equilibrium average volume V, which defines $< \rho >= N/ < V >$. The process is repeated for different P.

grand-canonical ensemble Here μ, T, V are fixed, and so one would choose a fixed μ, T, and V, calculate $< P >$ through the virial equation. ρ follows from from the equilibrium average $< N > /V$. The disadvantage here is that both $< P >$ and $< \rho >$ now have error bars. Another difficulty with performing grand-canonical simulations is that the probability to insert a particle becomes very low for a dense liquid. Special "biasing" techniques are needed to speed up the simulations. An advantage of this ensemble is that the chemical potential μ, from which many other properties can be calculated, automatically follows from the simulation.

The upshot of all this is simply that the optimum choice of ensemble depends very much on what properties one wants to investigate. This can be particularly important when studying phase-transitions.

2.4 Widom insertion trick

The chemical potential often seems mysterious when it is first introduced in statistical mechanics. But its meaning becomes more intuitive when you use a clever method, first introduced by Benjamin Widom[9], to calculate it using MC. Within the canonical ensemble, the chemical potential is defined as

$$\mu = \left(\frac{\partial A}{\partial N}\right)_{V,T} \tag{2.15}$$

where $A(N,V,T)$ is the Helmholtz free energy defined as $\beta A = -\log[Q(N,V,T)]$, with $Q(N,V,T)$ the canonical partition function. In the limit of a very large number of particles, the derivative of Eq. (2.15) can be estimated as

$$\mu = -k_B T \log\left[\frac{Q_{N+1}}{Q_N}\right]. \tag{2.16}$$

If we rewrite the partition function for a liquid (indirectly defined through Eq. (1.22)) in terms of scaled coordinates (similar to what was done for Eq. (2.7)), then the ratio of the two partition functions in Eq. (2.16) can be rewritten as:

$$\begin{aligned} \mu &= -k_B T \log\left[\frac{V}{\Lambda^3(N+1)}\right] - k_B T \log\left[\frac{\int d\mathbf{s}^{N+1} \exp\left[-\beta\mathcal{V}(\mathbf{s}^{N+1})\right]}{\int d\mathbf{s}^{N} \exp\left[-\beta\mathcal{V}(\mathbf{s}^{N})\right]}\right] \\ &= \mu_{id} + \mu_{ex}. \end{aligned} \tag{2.17}$$

$\mu_{id} = -k_B T \log[\rho\Lambda^3]$ is the known chemical potential of an ideal gas (using $V/(N+1) \approx \rho$). By separating out the coordinates $\mathbf{s}_{N+1}$ of particle $N+1$, and writing their effect on the potential energy as $\mathcal{V}(\mathbf{s}^{N+1}) = \mathcal{V}(\mathbf{s}^N) + \Delta\mathcal{V}$, μ_{ex} can be expressed as:

$$\begin{aligned} \mu_{ex} &= -k_B T \log\left[\int d\mathbf{s}_{N+1} \left(\frac{\int d\mathbf{s}^N \exp\left[-\beta\Delta\mathcal{V}\right] \exp\left[-\beta\mathcal{V}(\mathbf{s}^N)\right]}{\int d\mathbf{s}^N \exp\left[-\beta\mathcal{V}(\mathbf{s}^N)\right]}\right)\right] \\ &= -k_B T \log\left[\int d\mathbf{s}_{N+1} \left\langle \exp\left[-\beta\Delta\mathcal{V}\right]\right\rangle_{NVT}\right] \end{aligned} \tag{2.18}$$

where $<>_{NVT}$ denotes a canonical average over an N particle system. In other words, the excess chemical potential has been rewritten in terms of the ensemble average of the Boltzmann factor for inserting an extra particle into an N particle system. Since the average is over the original N particle system, this additional particle at $\mathbf{s}_{N+1}$ does not perturb the configuration $\mathbf{s}^N$. In other words, it is a "ghost" particle, whose only purpose is to "measure" the excess Boltzmann factor in Eq. (2.18). To implement this into a MC code, you attempt trial insertions, monitor the average of the Boltzmann factor, but never actually add the particle to the system.

[9] B. Widom, J. Chem. Phys. **39**, 2808 (1963)

subroutine MCWidom	Widom insertion
*xi = (ran(iseed)-0.5) * delL$_x$*	pick a random x coordinate
*yi = (ran(iseed)-0.5) * delL$_y$*	pick a random y coordinate
*zi = (ran(iseed)-0.5) * delL$_z$*	pick a random z coordinate
call energy(xi,yi,zi,Ei)	energy Ei of adding the particle
*mutest := mutest + exp(-beta * Ei)*	add to Boltzmann average
return	
end	

The Widom insertion trick provides an intuitive interpretation of the chemical potential as a measure of the difficulty of "adding" an excess particle to the system. In practise the simple sampling scheme described above begins to break down for dense fluids, where most insertions overlap with other particles, and thus have a very small Boltzmann weight. The average is then dominated by those rare insertions where the particle doesn't have overlaps, leading to poor statistics. Another subtlety to watch out for is that the excess chemical potential shows fairly important finite size effects:

$$\mu_{ex} = \langle \mu_{ex} \rangle_{N,V,T} + \mathcal{O}(\frac{1}{N}) \tag{2.19}$$

i.e. the prefactor in front of the $1/N$ factor is large. Remember that these finite size effects are systematic errors – they are not the same as statistical fluctuations!

2.5 Thermodynamic integration

Simulations are often employed to study first order phase-transitions, such as the freezing of a liquid. A naive way to investigate this liquid to solid transition would be to take a liquid in a simulation, lower the temperature, and simply wait for the system to freeze. In practise this is not usually a very good idea for the following reasons: Firstly, the *dynamics* of the phase-transition may be very slow, i.e. you may have to wait for a very long time indeed before you see anything. Remember that the (microscopic) timescales of your simulation are extremely short compared to what you might routinely see in a real life experiment[10]. Secondly, when the system starts to freeze, it first needs to form an interface. The free-energy of the interface scales with the area A of the interface, and so is negligible in the thermodynamic limit. However, for a finite sized simulation, this energy may not be negligible at all. Take, for example, a box with 1000 atoms in it, and let's assume that the fluid-solid interface is only two particles thick. There are typically 10 atoms along a side, and so a planar interface would contain about 200 atoms, i.e. 20% of the total, leading to clearly noticeable effects.

The obvious way forward would be to separately calculate the free-energies of the liquid and solid phases, A_l and A_s respectively, and then use a common tangent or other standard

[10]Strictly speaking, of course, MC has no physical timescale, since you are randomly walking through state space. By contrast, in MD there is a well-defined physical time, and total simulation times are typically on the order of a few nanoseconds. In practise, one can still define an effective MC timescale related to the number of independent parts of state-space covered during a simulation. A common approach is to define a "MC time" by attributing a certain physical time to each single particle move. This sometimes works rather well, but a simple relationship between the number of moves and time breaks down when clever MC techniques, which would correspond to non-physical moves, are employed.

thermodynamic construction to derive the phase-transition lines. But again, this is easier said than done, because free-energies are direct measures of the partition function, $\beta A = -\ln Q(N,V,T)$, which is very hard to calculate, as discussed in detail in chapter 1.

One way around this problem is to calculate the *difference* in free-energy between the system you are interested and some reference system for which the free-energy is known (like the ideal gas or the harmonic solid). To do this define a path

$$V(\lambda) = V_{ref} + \lambda \left(V_{sys} - V_{ref}\right) \tag{2.20}$$

between the Hamiltonian of the system, with potential energy function $\mathcal{V}_{sys} = V(\lambda = 1)$ and the Hamiltonian of the reference system, defined by $\mathcal{V}_{ref} = V(\lambda = 0)$ (The dependence on $\mathbf{r}^N$ has been suppressed for notational clarity. Remember also that the kinetic parts are the same). Here we've chosen a linear path, but that isn't necessary, although it has some important advantages. The partition function for arbitrary λ is given by:

$$Q(N,V,T;\lambda) = \frac{1}{\Lambda^{3N} N!} \int d\mathbf{r}^N \exp\left[-\beta V(\lambda)\right]. \tag{2.21}$$

Taking the derivative of the free-energy $\partial A(\lambda)/\partial\lambda = -\partial \log\left[Q(N,V,T;\lambda)\right]/\partial\lambda$, brings down one factor of $V(\lambda)$:

$$\left(\frac{\partial A(\lambda)}{\partial \lambda}\right) = \frac{\int d\mathbf{r}^N \left(\partial V(\lambda)/\partial\lambda\right) \exp\left[-\beta V(\lambda)\right]}{\int d\mathbf{r}^N \exp\left[-\beta V(\lambda)\right]} \quad = \quad \left\langle \frac{\partial V(\lambda)}{\partial \lambda} \right\rangle_\lambda, \tag{2.22}$$

where as usual the $\Lambda^{3N} N!$ terms cancel. The average $< \ldots >_\lambda$ can be viewed as ensemble average of $\partial V(\lambda)/\partial\lambda$ over a system interacting with the potential $V(\lambda)$.

The total free-energy follows from a simple integration:

$$A(\lambda = 1) - A(\lambda = 0) = \int_{\lambda=0}^{\lambda=1} d\lambda \left\langle \frac{\partial V(\lambda)}{\partial \lambda} \right\rangle_\lambda. \tag{2.23}$$

Since the reference free-energy $A(\lambda = 0)$ is known, $A(\lambda = 1)$ has been found without needing a direct calculation of the partition function.

In practise you would perform a number of simulations at different λ, each with an interaction given by $V(\lambda)$ of Eq. (2.20), and then numerically integrate Eq. (2.23). How many different simulations you need depends partially on how Eq. (2.23) varies with λ. For many applications this number is small, on the order of 10 or less.

In this way you can calculate the free energy of the system under investigation. To calculate something like the location of a freezing transition, you would need the free energy at a number of different state points. For each state point, you would need to do the thermodynamic integration. In other words, calculating exactly where a freezing transition occurs can be quite a lot of work.

Chapter 3

More advanced methods

The number of more complex MC techniques – tailored to all manner of different physical problems – is enormous, and growing rapidly. Since these special MC methods can achieve important speedups in simulation efficiency, performing a literature search before you embark on a new MC simulation project is usually time well spent. This chapter provides a small taster of two classes of advanced techniques: 1) Methods to increase the sampling efficiency in statistical mechanics and 2) Methods to treat quantum mechanical problems.

3.1 Clever sampling techniques increase MC efficiency

Up until now we have mainly used the simple Metropolis prescription of of Eq. (1.21), which was derived by assuming that the transition matrices to choose a trial step from $i \to j$ are symmetric, i.e. $\alpha(i \to j) = \alpha(j \to i)$. With this simplification, imposing the detailed balance condition of Eq. (1.18) on the total transition probability $\pi(i \to j) = \alpha(i \to j)acc(i \to j)$ only fixes the ratio of the acceptance probabilities. The Metropolis algorithm fulfils this using the recipe $acc(i \to j) = min\{1, P(i)/P(j)\}$, where the distribution $P(i)$ was taken to have the form $P(E_i) \propto \exp\left[-\beta\mathcal{V}(\mathbf{r}_i^N)\right]$ for the canonical (NVT) ensemble, or something similar for the constant pressure or grand-canonical ensembles. Defining a more general acceptance parameter χ such that $acc(i \to j) = min\{1, \chi\}$ and $acc(j \to i) = min\{1, 1/\chi\}$, leads to the following detailed balance condition:

$$P(i)\alpha(i \to j)min\{1, \chi\} = P(j)\alpha(j \to i)min\left\{1, \frac{1}{\chi}\right\} \tag{3.1}$$

which determines χ:

$$\chi = \frac{P(j)\alpha(j \to i)}{P(i)\alpha(i \to j)} \tag{3.2}$$

With this recipe we can easy change:

1. The transition matrices $\alpha(i \to j)$, which determine the probability to select a particular kind of move.

2. The probability distribution $P(i)$ over which to sample.

and then use Eq. (3.2) to derive the correct acceptance probability that satisfies detailed balance.

Changes of type 1 are often useful when we know beforehand that some particular states are important for our averages, but infrequently sampled by just random particle moves. An example of this is given by the section on association bias MC.

We already applied changes of type 2 when using different ensembles, but there are some cases where, even though the system is described by a particular ensemble, we may still want to sample over a different distribution. Say you are working in the canonical ensemble, where averages are taken over a distribution proportional to $\exp\left[-\beta E_i\right]$, but you want to generate your MC chain according to a non-Boltzmann distribution:

$$P_{nB}(i) \propto \exp\left[-\beta\left(E_i + \Delta E_i\right)\right]. \tag{3.3}$$

The normal partition function can be rewritten

$$\begin{aligned} Q &= \sum_i \exp\left[-\beta E_i\right] = \frac{Q_{nB}}{Q_{nB}} \sum_i \exp\left[-\beta\left(E_i + \Delta E_i\right)\right] \exp\left[\beta \Delta E_i\right] \\ &= Q_{nB} \left\langle \exp\left[\beta \Delta E_i\right]\right\rangle_{nB} \end{aligned} \tag{3.4}$$

where $\langle \ldots \rangle_{nB}$ is an average over the non-Boltzmann distribution of Eq. (3.3), and $Q_{nB} = \sum_i \exp\left[-\beta(E_i + \Delta E_i)\right]$. Canonical ensemble averages can also be rewritten in a similar way:

$$\begin{aligned} < A > &= \frac{1}{Q} \sum_i A_i \exp\left[-\beta E_i\right] \\ &= \frac{1}{Q_{nB}} \sum_i \left(A_i \exp\left[\beta \Delta E_i\right] \exp\left[-\beta\left(E_i + \Delta E_i\right)\right]\right) \left(\frac{Q_{nB}}{Q}\right) \\ &= \left\langle A \exp\left[\beta \Delta E_i\right]\right\rangle_{nB} / \left\langle \exp\left[\beta \Delta E_i\right]\right\rangle_{nB} \end{aligned} \tag{3.5}$$

Note the resemblance to the importance sampling of 1-d integrals of Eq. (1.7).

Why sample over non-Boltzmann distributions that differ from the statistical mechanical probability to find a given state? The reason typically has to do with quasi-ergodicity problems, where the system is stuck in one part of phase-space and can't easily get to another: for example, if there is a bottleneck, as depicted schematically in Fig. 3.1. Say we were performing a MC simulation on a simple two-well energy landscape as shown in Fig. 1.6. One way of regularly making it over the barrier, even if its height $\beta E_b >> 1$, would be to add a biasing potential which is of the form $\Delta E \approx -E_b$ in the barrier region, but zero in the two wells. Trajectories based on this potential would waste some simulation time in the barrier region, but on the other hand the system could easily move between the two wells, and thus solve the quasi-ergodicity problem[1].

Adding a special biasing potential ΔE only works well if we can make a good a-priori guess of where the bottlenecks are. For a more complex energy landscape, this rapidly becomes prohibitively difficult. An alternative form of non-Boltzmann sampling, called "parallel tempering" circumvents this problem by using trajectories at a higher temperature, which move more easily from one minimum to another. The coordinates of the system of interest are occasionally switched with the higher temperature simulation, leading to a more rapid exploration of phase-space. Details are described in the next section.

[1]This class of non-Boltzmann sampling techniques is know as "umbrella sampling". A nice pedagogical example can be found in the book by David Chandler, "Introduction to Modern Statistical Mechanics" OUP, (1987), chapter 6

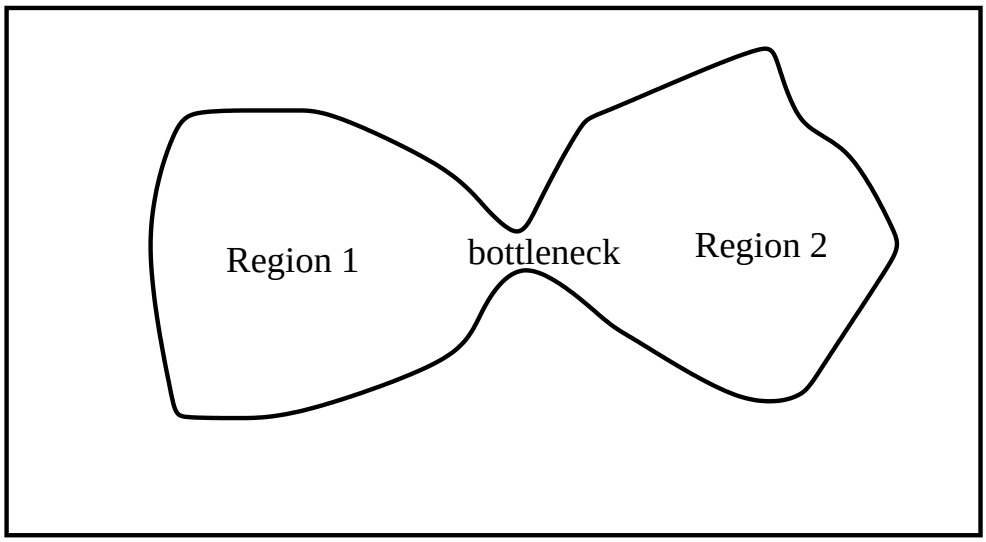


Figure 3.1: A schematic picture of a bottleneck between the two important regions of phase-space. This could represent a system like that of Fig.1.6 when the two energy minima aren't that different in energy, but the barrier is much higher than the effective temperature of the simulation. Regions 1 and 2 above correspond to the two wells, and the bottleneck is caused by the potential barrier between them.

3.1.1 Parallel tempering

There are many interesting physical systems whose free-energy landscape has a form similar to the one depicted schematically in Fig. 3.1.1 – the wells are separated by large barriers. If you are interested in temperature much lower than the barrier heights, then a standard MC simulation will become very inefficient: it will spend most of its time in only a few wells, instead of sampling the whole (free) energy surface. Well known examples of such systems include structural glasses and the conformations of proteins in solution. However, if you were interested in a temperature much higher than the average barrier height, you would have no such ergodicity problems; your system would happily move from energy minimum to energy minimum. This leads to the idea of parallel tempering: instead of performing one simulation at temperature T, do a series of simulations in parallel, at different temperatures, and occasionally swap the conformations of one system with another. In this way the low temperature simulation can make use of the fact that the high temperature MC trajectories sample many different minima.

To implement this intuitive and very powerful scheme, we first need to derive the transition matrices and acceptance criteria that satisfy detailed balance, which is done in the next box.

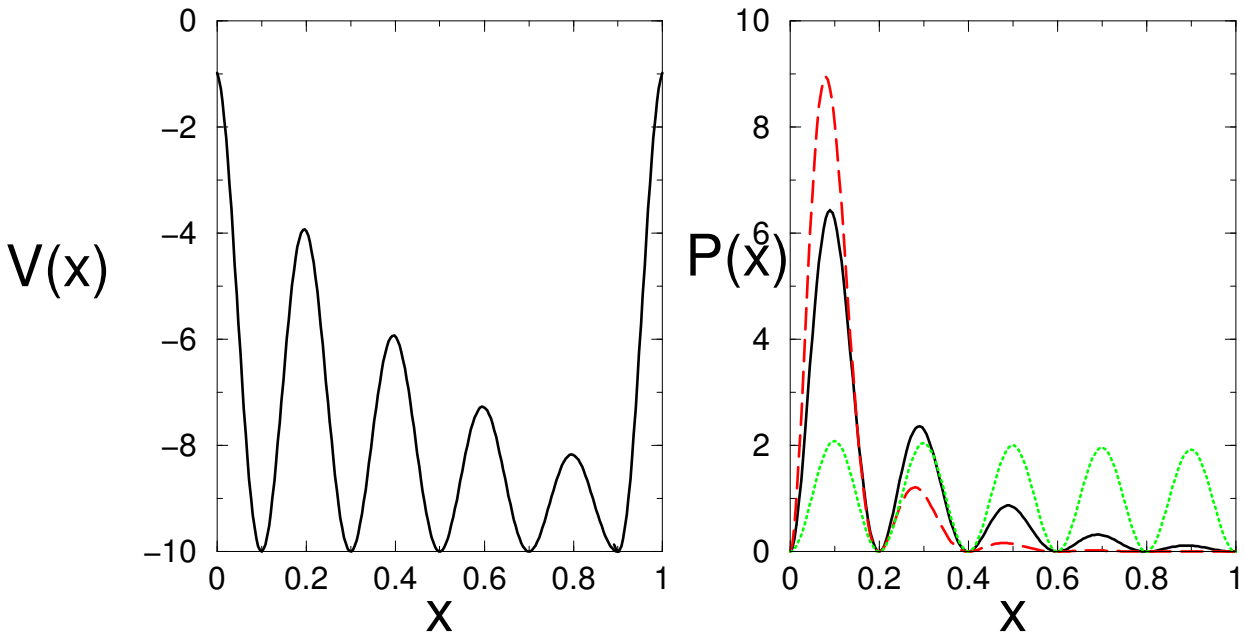


Figure 3.2: The graph on the left depicts a model energy land-scape. At every temperature, the probability of being in any well should be equal. The graph on the right shows what the normalised distribution $P(x)$ might look like after a finite (but large) number of MC steps, when the simulation is started in the left most well. For a simulation at the lowest temperature (dashed line), the MC trajectory is likely to stay stuck in the first basin. At the highest temperature (dotted line), the system easily moves from one basin to another. (For an infinite amount of time all three distributions would indeed look similar – this is really a quasi-ergodicity problem, something very common in MC simulations.

Transition matrix and acceptance probability for a parallel tempering move

We will consider parallel tempering moves that swap the temperature (or equivalently the configurations) between a set of M simultaneous canonical MC simulations, each at a temperature T_k, and described by a partition function $Q(NVT_k)$.

Transition matrix

The transition matrix is fairly easy to determine – just choose two of the M systems at random and switch the temperatures. Since all switches are equally likely to be chosen (although not accepted of course), the transition matrix is symmetric, which simplifies the derivation of the correct acceptance probabilities.

Acceptance probability

To analyse the acceptance probability of parallel tempering MC move it is useful to define an *extended ensemble* of all M systems:

$$Q_{extended}(N,V,\{T_k\}) = \prod_{k=1}^{M} Q(NVT_k) = \prod_{k=1}^{M} \frac{1}{\Lambda_k^{3N} N!} \int d\mathbf{r}_k^N \exp\left[-\beta_k \mathcal{V}\left(\mathbf{r_k^N}\right)\right] \tag{3.6}$$

where each system has its own set of particle coordinates $\mathbf{r_k^N}$. Suppose we choose to attempt a switch of temperature (or equivalently β) between two simulations, a and b, drawn from the M different systems. To satisfy detailed balance in the extended ensemble we require:

$$\begin{aligned} P(\mathbf{r_a^N},\beta_\mathbf{a})\mathbf{P}(\mathbf{r_b^N},\beta_\mathbf{b}) \quad &\times \quad acc\left(\left\{(\mathbf{r_a^N},\beta_\mathbf{a}),(\mathbf{r_b^N},\beta_\mathbf{b})\right\} \rightarrow \left\{(\mathbf{r_a^N},\beta_\mathbf{b}),(\mathbf{r_b^N},\beta_\mathbf{a})\right\}\right) = \\ P(\mathbf{r_a^N},\beta_\mathbf{b})\mathbf{P}(\mathbf{r_b^N},\beta_\mathbf{a}) \quad &\times \quad acc\left(\left\{(\mathbf{r_a^N},\beta_\mathbf{b}),(\mathbf{r_b^N},\beta_\mathbf{a})\right\} \rightarrow \left\{(\mathbf{r_a^N},\beta_\mathbf{a}),(\mathbf{r_b^N},\beta_\mathbf{b})\right\}\right) \end{aligned} \tag{3.7}$$

where we have made use of the fact that the transition matrices are symmetric, and therefore cancel on both sides of Eq. (3.7). By using the extended canonical ensemble of Eq. (3.6), the ratio of the two acceptance probabilities simplifies to

$$\begin{aligned} \chi \quad &= \quad \frac{acc\left(\left\{(\mathbf{r_a^N},\beta_\mathbf{a}),(\mathbf{r_b^N},\beta_\mathbf{b})\right\} \rightarrow \left\{(\mathbf{r_a^N},\beta_\mathbf{b}),(\mathbf{r_b^N},\beta_\mathbf{a})\right\}\right)}{acc\left(\{(\mathbf{r_a^N},\beta_\mathbf{b}),(\mathbf{r_b^N},\beta_\mathbf{a})\} \rightarrow \{(\mathbf{r_a^N},\beta_\mathbf{a}),(\mathbf{r_b^N},\beta_\mathbf{b})\}\right)} \\ &= \frac{P(\mathbf{r_a^N},\beta_\mathbf{b})\mathbf{P}(\mathbf{r_b^N},\beta_\mathbf{a})}{P(\mathbf{r_a^N},\beta_\mathbf{a})\mathbf{P}(\mathbf{r_b^N},\beta_\mathbf{b})} \\ &= \frac{\exp\left[-\beta_b \mathcal{V}(\mathbf{r_a^N}) - \beta_\mathbf{a}\mathcal{V}(\mathbf{r_b^N})\right]}{\exp\left[-\beta_a \mathcal{V}(\mathbf{r_a^N}) - \beta_\mathbf{b}\mathcal{V}(\mathbf{r_b^N})\right]} \\ &= \exp\left[(\beta_a - \beta_b)\left(\mathcal{V}(\mathbf{r_a^N}) - \mathcal{V}(\mathbf{r_b^N})\right)\right] \end{aligned} \tag{3.8}$$

So the acceptance criterion in parallel tempering reduces to a fairly simple form: $acc(0 \rightarrow n) = min\{1, \exp(\Delta\beta\Delta V)\}$, with $\Delta\beta$ the change in temperature, and ΔV the change in potential energy.

To implement a parallel tempering simulation, we need to run the M different systems simultaneously, using some set of standard single system MC moves. The probability to choose a parallel tempering move should be adjusted to maximise sampling efficiency. This will depend very much on details of the simulation. These switching moves are usually not expensive to

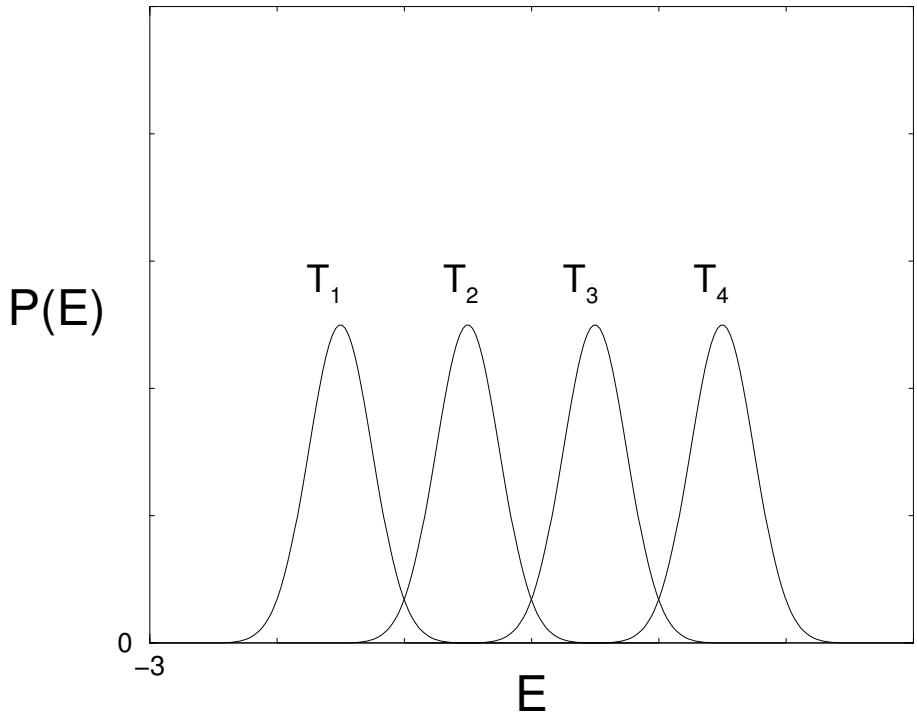


Figure 3.3: The probability to find a configuration with a certain energy E changes with temperature T_k. States in the overlap between two distributions are the most likely be accepted during a parallel tempering move. Adding intermediate temperatures can greatly increase the rate at which swaps are accepted.

calculate, since the $\beta_k V(\mathbf{r_k^N})$ are already known for each system. However, if the difference in temperature is large, then the two systems will tend to explore different areas of phase-space, and the acceptance probability might be very low. This is illustrated in Fig. 3.1.1. Running many intermediate temperatures increases the probability of parallel tempering switches, but comes at the cost of running extra simulations. The optimum temperature increment is the one that leads to maximum overall computational efficiency.

The idea of parallel tempering is quite general. Other variables can also be switched. For example, when calculating a phase-diagram in the grand-canonical ensemble, you can switch the chemical potential of different systems to achieve important MC speedups. Similarly, when performing thermodynamic integration, you need to simulate at differing Hamiltonians. If all the simulations, from the reference system to the system of interest, have similar MC efficiencies, there would be no advantage in performing parallel tempering moves. However, if some of the intermediate Hamiltonians exhibit quasi-ergodicity problems, parallel tempering could speed things up. A number of other applications of parallel tempering, including finding zeolite structures from powder diffraction data, and simulating different lengths of polymers, are described in FS2002.

3.1.2 Association-Bias Monte Carlo

Suppose that we want to simulate a system where the density is very low, but the interparticle attraction is strong, so that the particles easily form clusters, as shown in Fig. 3.1.2. The

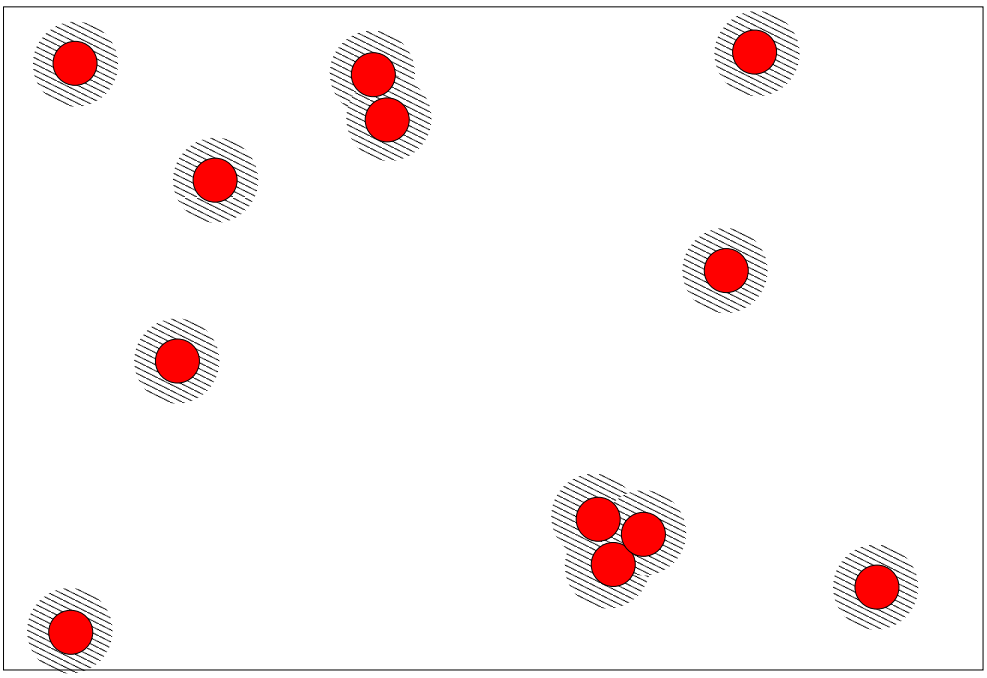


Figure 3.4: A system with strong interparticle attractions but a low density will tend to form clusters. Energy favours the formation of clusters, while entropy favours their break-up, so that at equilibrium there is a distribution of cluster sizes. Standard MC moves inefficiently sample phase-space (can you see why?). Adding cluster association moves, which preferentially place a particle in the bonding region V_a where it feels the attraction of another molecule (depicted by the shaded areas in the figures), greatly increases the efficiency of the MC simulation. To satisfy detailed balance these cluster association moves must have a counterpart that breaks up the clusters.

standard MC algorithm with random displacements would be very inefficient because the chance of such a random move bringing two particles coming close enough to form a cluster is extremely small. In other words, most particle moves would not change the overall energy $\mathcal{V}(\mathbf{r}^N)$, whereas the thermodynamic averages can be dominated by terms where $\exp\left[-\beta\mathcal{V}(\mathbf{r^N})\right]$ is large.

One way to speed up the MC code would be to add cluster association moves, biased towards bringing particles close together. To satisfy detailed balance, there must be moves that break the clusters up as well. This is worked out in more detail in the next box.

When performing a MC simulation one chooses, with a particular probability, between cluster moves (association or breakup) and ordinary random displacement MC moves. Their relative frequency depends on details of the system, and should be adjusted to maximise efficiency. To satisfy detailed balance, it is important that the choice between these two classes of moves is random, and not sequential.

Cluster association moves

Proceed in two steps:
1) Choose a particle at random: the probability = $1/N$

2) Move it from $\mathbf{r_o}$ to a random trial position $\mathbf{r_n}$, constrained to be inside the volume V_a defined as the union of all the bonding regions around each molecule: the probability = $1/V_a$.

The total transition probability matrix $\alpha_a(o \to n)$ to make an such association move is:

$$\alpha_a(o \to n) = \frac{1}{NV_a} \tag{3.9}$$

Note that since the particle should be place uniformly within the bonding region, there is a finite probability of particle overlap (and thus rejection). However, the chance of making a successful cluster association move should still be much larger than the chance of moving the particle into a bonding region by a random displacement move.

Cluster breakup moves

1) Choose a particle at random from the N_a associated particles: the probability is $1/N_a$.

2) Move this particle to a random trial position $\mathbf{r_n}$: the probability = $1/V$

The total transition probability matrix $\alpha_b(o \to n)$ to make an cluster breakup move is:

$$\alpha_b(o \to n) = \frac{1}{N_a V} \tag{3.10}$$

Acceptance probabilities from detailed balance

We're still sampling from the Boltzmann distribution, so the $P(i)$ are known, while the transition matrices (which are now no longer symmetric!) are given by the two equations above. The recipe of Eq. (3.2) then defines the acceptance probability for a cluster association moves:

$$\chi = \frac{V_a N}{N_a V} \exp\left[-\beta\left(\mathcal{V}(\mathbf{r_i^N}) - \mathcal{V}(\mathbf{r_j^N})\right)\right] \tag{3.11}$$

The probability to accept a given cluster association move is therefore $acc_a(i \to j) = min\{1, \chi\}$, while for the reverse breakup move $acc_n(j \to i) = min\{1, 1/\chi\}$. In this way you satisfy detailed balance and generate your averages according to a Boltzmann distribution.

The Boltzmann factor can be much larger for a cluster association move than for a breakup move. For maximum efficiency we want roughly $\frac{1}{2}$ of both kinds of moves to be accepted; this can achieved by adjusting the size of the bonding region.

In practise, determining the total association volume V_a is rather expensive, since it changes whenever particles form a cluster (due to overlaps of the volume around each individual particle). However, some recent extensions to the simple ideas above seem to have solved this problem (see e.g. S. Wierzchowski and D. Kofke, J. Chem. Phys. **114**, 8752 (2001))

3.2 Quantum Monte Carlo techniques

Another field where Monte Carlo techniques are becoming increasingly important is the calculation of quantum mechanical properties. Some of these ideas go back to Fermi, Metropolis, and some of the others who were working together in Los Alamos when the very first Monte Carlo codes were developed. The most popular methods in use today are

- **Variational Monte Carlo**, where a trial wave-function is optimised according to the variational principle. This method is probably the easiest to understand, and will be the only one discussed in more detail in this course. Its advantages are that it is very versatile, and easy to implement and interpret. The downside is that you are limited in accuracy by the form of the variational wave-function.

- **Projector Monte-Carlo**, where a projection operator is repeatedly applied to a wave-function until only the ground state remains. In principle this method leads to exact ground state wave-functions. It has been used with some success to perform benchmark calculations on various (small) atomic and molecular systems.

- **Path-Integral Monte Carlo**. This beautiful method exploits the isomorphism between quantum-mechanics and the statistical mechanics of ring polymers[2] By using standard MC algorithms to calculate the properties of the (classical) ring polymers – where each bead corresponds to a particle at a different slice of "imaginary time" – one finds the equilibrium properties of a quantum system at finite temperature.

Each method has its advantages and disadvantages. For a nice overview I recommend the web site of David Ceperley: http://archive.ncsa.uiuc.edu/Science/CMP/method.html

3.2.1 Variational Monte Carlo

The only method we will discuss in some detail is variational MC (VMC). As its name suggests, it is based on the variational theorem of quantum mechanics, which states that for any trial wave-function Ψ, appropriate to the Hamiltonian $\mathcal{H}$, the variational energy E_V, given by

$$E_v = \frac{\langle \Psi \left| \mathcal{H} \right| \Psi \rangle}{\langle \Psi | \Psi \rangle} \geq E_0, \tag{3.12}$$

is always greater than the ground state energy E_0. This can be rewritten as:

$$E_v = \frac{\int d\mathbf{r} |\mathbf{\Psi}(\mathbf{r})|^2 \left[\frac{\mathcal{H}\mathbf{\Psi}(\mathbf{r})}{\mathbf{\Psi}(\mathbf{r})} \right]}{\int d\mathbf{r} |\mathbf{\Psi}(\mathbf{r})|^2} = \int P(\mathbf{r}) \epsilon(\mathbf{r}) \tag{3.13}$$

where the positive sampling distribution over the generalised coordinates $\mathbf{r}$ is given by

$$P(\mathbf{r}) = \frac{|\mathbf{\Psi}(\mathbf{r})|^2}{\int \mathbf{dr} |\mathbf{\Psi}(\mathbf{r})|^2} \tag{3.14}$$

[2]Quantum Mechanics can be completely rewritten in terms of path-integrals, see e.g. the classic book by R.P. Feynman and A.R. Hibbs *Quantum Mechanics and Path Integrals*, McGraw-Hill, New York (1965)

and the "local" energy is defined as:

$$\epsilon(\mathbf{r}) = \frac{\mathcal{H}\Psi(\mathbf{r})}{\Psi(\mathbf{r})} \tag{3.15}$$

There is a direct analogy with classical MC: $P(\mathbf{r})$ is a probability distribution over which an operator $\epsilon(\mathbf{r})$ is averaged. Therefore, the standard Metropolis recipe can be used. Trial moves correspond to random displacements in the generalised coordinate space $\mathbf{r}$. The acceptance probability $acc(o \rightarrow n) = min\left\{1, |\Psi(\mathbf{r_n})|^2/|\Psi(\mathbf{r_o})|^2\right\}$, results in a MC trajectory that samples space according to the probability $P(\mathbf{r})$. The average of $\epsilon(\mathbf{r})$ will converge to E_V with an error proportional to $\frac{1}{\sqrt{M}}$, where M is the number of independent steps in your MC chain[3].

The quality of E_v itself depends on the quality of your trial function. A particularly nice property of VMC is that if you choose the ground state wave-function $\Psi_0(\mathbf{r})$ as your trial function, then $\epsilon(r) = E_0$ for all r, and so the MC simulation has zero variance (Can you see why this is so? Plug a ground state into Eq. (3.13).). In other words, choosing a good trial function both brings E_v closer to E_0, *and* leads to a smaller statistical error in the MC sampling. Choosing good variational functions is therefore particularly important in VMC.. On the one hand, the wave-function should not be so complicated that the determination of $P(\mathbf{r})$ and $\epsilon(\mathbf{r})$ are unduly expensive. On the other hand, we want it as close to the ground state as possible.

In-class exercise: Variational wave-function for a two-electron atom.

[3]For simple enough systems the variational integral of Eq. (3.12) could be calculated by normal quadrature. However, as we saw in chapter 1, this rapidly becomes intractable for integrals in higher dimensions, and MC techniques become relatively more efficient.

Part III Course M10

COMPUTER SIMULATION METHODS IN CHEMISTRY AND PHYSICS

Michaelmas Term 2005

SECTION 2: MOLECULAR DYNAMICS TECHNIQUES

Chapter 4

Basic molecular dynamics algorithm

4.1 Integrating the equations of motion

The aim of Molecular Dynamics is to study a system by recreating it on the computer as close to nature as possible, i.e by simulating the dynamics of a system in all microscopic detail over a physical length of time relevant to the properties of interest. The connection of MD to Statistical Mechanics involves basically the same questions as the relation of Statistical Mechanics to experiment. MD is, therefore, a good starting point for an introduction to numerical simulation.

4.1.1 Newton's equations of motion

We begin by recalling the basic microscopic dynamics underlying all (classical) statistical mechanics, namely Newton's equations of motion. We also introduce some of the notation that will be used throughout these notes. The Cartesian position vectors $\mathbf{r}_i$ of a system of N particles $i = 1, \cdots N$ will be collectively denoted by $\mathbf{r}^N$. The potential $\mathcal{V}$ specifying the interactions between the particles is completely determined by giving the positions of the particles. This dependence on atomic configuration is expressed in our notation as

$$\mathcal{V} = \mathcal{V}(\mathbf{r}_1, \mathbf{r}_2, ..\mathbf{r}_N) \equiv \mathcal{V}(\mathbf{r}^N) \tag{4.1}$$

The force $\mathbf{f}_i$ on particle i is obtained as a partial derivative of $\mathcal{V}$ and is, of course, equally a function of configuration $\mathbf{r}^N$.

$$\mathbf{f}_i\left(\mathbf{r}^N\right) = -\frac{\partial \mathcal{V}\left(\mathbf{r}^N\right)}{\partial \mathbf{r}_i} \tag{4.2}$$

Newton's equations of motion for our N particle system are written as a set of N coupled second order differential equations in time.

$$m_i \ddot{\mathbf{r}}_i = \mathbf{f}_i\left(\mathbf{r}^N\right) \tag{4.3}$$

where m_i is the mass of particle m_i. A quantity often used in formal derivations in dynamics and statistical mechanics is the momentum. For Cartesian coordinates the momentum of particle i is simply proportional to its velocity.

$$\mathbf{p}_i = m_i \dot{\mathbf{r}}_i \tag{4.4}$$

Interatomic interactions, as opposed to the forces related to chemical bonds keeping molecules together, are relatively weak. A good first approximation for interatomic interactions is that they are pair-wise additive. Moreover, for atoms these pair potentials can be assumed to be

central, i.e. depending only on distance. The total potential $\mathcal{V}$ can then be resolved in a sum of potentials v which are a function of a single variable, namely the length r_{ij} of the vector measuring the displacement $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$ of particle i with respect to particle j.

$$\mathcal{V}(\mathbf{r}^N) = \sum_{i=1}^{N} \sum_{j=i+1}^{N} v(r_{ij}) \tag{4.5}$$

In the summation we have made sure that every pair of atoms i, j is only counted once by imposing a condition $j > i$. This restriction can be lifted by using that the length of a vector is the same if we interchange begin and endpoint $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j| = |\mathbf{r}_j - \mathbf{r}_i| = r_{j\,i}$ and therefore also $v(r_{ij}) = v(r_{ji})$. Hence we can write

$$\mathcal{V}(\mathbf{r}^N) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} v(r_{ij}) \tag{4.6}$$

Now every pair is counted twice, but the overcounting is corrected with by the factor 1/2. Of course, self interaction ($i = j$) is still excluded. Summation starting from a lower limit 1 is considered default and is usually suppressed. With this convention Eq. 4.6 can be written as

$$\mathcal{V}(\mathbf{r}^N) = \frac{1}{2} \sum_{i}^{N} \sum_{j\neq i}^{N} v(r_{ij}) \tag{4.7}$$

with the condition $j \neq i$ on the values of index j stated below the corresponding summation sign. Expression Eq. 4.6 makes it easier to see that also the force on particle i is a superposition of pair forces $\mathbf{f}_{ij}$

$$\mathbf{f}_i = -\frac{\partial \mathcal{V}}{\partial \mathbf{r}_i} = -\frac{1}{2} \sum_{j\neq i}^{N} \left(\frac{\partial v(r_{ij})}{\partial \mathbf{r}_i} + \frac{\partial v(r_{j\,i})}{\partial \mathbf{r}_i} \right) = -\sum_{j\neq i}^{N} \frac{\partial v(r_{ij})}{\partial \mathbf{r}_i} = \sum_{j\neq i}^{N} \mathbf{f}_{ij} \tag{4.8}$$

Pair forces satisfy Newtons third law in detail

$$\mathbf{f}_{j\,i} = -\mathbf{f}_{ij} \tag{4.9}$$

To prove Eq. 4.9 we first apply the chain rule

$$\mathbf{f}_{ij} = -\frac{\partial v(r_{ij})}{\partial \mathbf{r}_i} = -\frac{\partial r_{ij}}{\partial \mathbf{r}_i} \left. \frac{dv(r)}{dr} \right|_{r=r_{ij}} \tag{4.10}$$

and then use the vector relation

$$\nabla r = \frac{\mathbf{r}}{r} \tag{4.11}$$

taking for $\mathbf{r} = \mathbf{r}_{ij} = -\mathbf{r}_{j\,i}$ we find

$$\frac{\partial r_{ij}}{\partial \mathbf{r}_i} = \frac{\mathbf{r}_{ij}}{r_{ij}} = -\frac{\mathbf{r}_{j\,i}}{r_{ij}} = -\frac{\partial r_{ij}}{\partial \mathbf{r}_j} \tag{4.12}$$

Substitution of Eq. 4.12 in Eq. 4.10 gives Eq. 4.9. The symmetry rule Eq. 4.9 is only valid for systems isolated from the outside world and has a number of important implications (see problem X). It is also very convenient in numerical computation.

4.1.2 Energy conservation and time reversal symmetry

A more fundamental property of mechanical systems, for pair or many body interactions provided they can be derived from a potential, is that total energy is conserved during the motion. Total energy $\mathcal{E}$ is the sum of the potential energy $\mathcal{V}$ and the kinetic energy $\mathcal{K}$

$$\mathcal{K} = \sum_i^N \frac{1}{2} m_i \dot{\mathbf{r}}_i^2 \tag{4.13}$$

Hence the quantity

$$E = \mathcal{K} + \mathcal{V} \equiv \mathcal{H} \tag{4.14}$$

must be a constant of motion, i.e its total time derivative is zero

$$\frac{dE}{dt} = 0 \tag{4.15}$$

The quantity introduced by the second identity of Eq. 4.14 is the Hamiltonian of the system, which, for the moment, is no more than another symbol for the total energy. To prove Eq. 4.15 we expand the total time derivative using the chain rule

$$\frac{dE}{dt} = \frac{d}{dt}\left[\sum_i^N \frac{\mathbf{p}_i^2}{2m_i} + \mathcal{V}\right] = \sum_i^N \frac{\mathbf{p}_i \cdot \dot{\mathbf{p}}_i}{m_i} + \sum_i^N \frac{\partial \mathcal{V}}{\partial \mathbf{r}_i} \cdot \dot{\mathbf{r}}_i \tag{4.16}$$

Substituting Newtons equation of motion Eq. 4.3 and the relations between velocity and momentum (Eq. 4.4) and force and potential (Eq. 4.2)

$$\frac{dE}{dt} = \sum_i^N \dot{\mathbf{r}}_i \cdot \mathbf{f}_i + \sum_i^N \frac{\partial \mathcal{V}}{\partial \mathbf{r}_i} \cdot \dot{\mathbf{r}}_i = -\sum_i^N \dot{\mathbf{r}}_i \cdot \frac{\partial \mathcal{V}}{\partial \mathbf{r}_i} + \sum_i^N \frac{\partial \mathcal{V}}{\partial \mathbf{r}_i} \cdot \dot{\mathbf{r}}_i = 0 \tag{4.17}$$

Conservation of energy is fundamental in the derivation of the equilibrium ensembles in statistical mechanics. It can also be applied as a very powerful test of the stability of numerical schemes for the integration of the equations of motion. We will repeatedly return to this point. Another formal feature of Newtonian dynamics that plays a role both in the theory of statistical mechanics and in the practice of the development of MD algorithms is time reversal symmetry. This rather mysterious principle states that if we reverse at a given time t all velocities, keeping the positions the same, the system will retrace its trajectory back into the past. In order to express this symmetry in a formal way we first make the dependence on initial condition explicit.

$$\mathbf{r}^N\left(t\,;\mathbf{r}_0^N,\mathbf{p}_0^N\right) \equiv \left\{ \begin{array}{l} \mathbf{r}^N(t) \\ \mathbf{r}^N(0) = \mathbf{r}_0^N,\ \mathbf{p}^N(0) = \mathbf{p}_0^N \end{array} \right. \tag{4.18}$$

In this notation, time reversal symmetry implies the relation

$$\begin{array}{rcl} \mathbf{r}^N\left(t;\,\mathbf{r}^N(0),-\mathbf{p}^N(0)\right) & = & \mathbf{r}^N\left(-t;\,\mathbf{r}^N(0),\mathbf{p}^N(0)\right) \\ \mathbf{p}^N\left(t;\,\mathbf{r}^N(0),-\mathbf{p}^N(0)\right) & = & -\mathbf{p}^N\left(-t;\,\mathbf{r}^N(0),\mathbf{p}^N(0)\right) \end{array} \tag{4.19}$$

On the leeft hand side of Eq. 4.19, $\mathbf{r}^N(0), \mathbf{p}^N(0)$ play the role of final conditions, which similar to initial conditions, also completely determine a trajectory.

4.1.3 The Verlet algorithm

Molecular dynamics methods are iterative numerical schemes for solving the equations of motion of the system. The first step is a discretization of time in terms of small increments called time steps which we will assume to be of equal length δt. Counting the successive equidistant points on the time axis by the index m, the evolution of the system is described by the series of the coordinate values

$$\cdots, \mathbf{r}^N(t_{m-1}) = \mathbf{r}^N(t_m - \delta t), \mathbf{r}^N(t_m), \mathbf{r}^N(t_{m+1}) = \mathbf{r}^N(t_m + \delta t), \cdots \tag{4.20}$$

plus a similar series of velocities $\dot{\mathbf{r}}^N$. Here, $\mathbf{r}^N$ again stands for the complete set of coordinates $\mathbf{r}_i, i = 1, \cdots N$. The MD schemes we will discuss all use the Cartesian representation. There are, in fact, good reasons for preferring this description for the chaotic interacting many body systems which are the subject of statistical mechanics.

An elementary integrator that has found wide application in MD is the Verlet algorithm. It is based on a clever combination of a Taylor expansion forward and backward in time. The third order approximation for $\mathbf{r}_i$ at time time $t + \delta t$ is

$$\mathbf{r}_i(t + \delta t) = \mathbf{r}_i(t) + \delta t\, \mathbf{v}_i(t) + \frac{\delta t^2}{2m_i}\mathbf{f}_i(t) + \frac{\delta t^3}{6}\mathbf{b}_i(t) + O(\delta t^4) \tag{4.21}$$

where we have used the familiar symbol $\mathbf{v}_i$ for the velocity $\dot{\mathbf{r}}_i$ of particle i and have inserted the equation of motion Eq. 4.3 in the second order term. If Eq. 4.21 and the corresponding approximation for as step backward in time

$$\mathbf{r}_i(t - \delta t) = \mathbf{r}_i(t) - \delta t\, \mathbf{v}_i(t) + \frac{\delta t^2}{2m_i}\mathbf{f}_i(t) - \frac{\delta t^3}{6}\mathbf{b}_i(t) + O(\delta t^4) \tag{4.22}$$

are added we obtain a prediction for $\mathbf{r}_i$ at the next point in time:

$$\mathbf{r}_i(t + \delta t) = 2\mathbf{r}_i(t) - \mathbf{r}_i(t - \delta t) + \frac{\delta t^2}{m_i}\mathbf{f}_i(t) + O(\delta t^4) \tag{4.23}$$

Note that the accuracy of the prediction is third order in time, i.e one order better than the Taylor expansion of Eqs. 4.21 and 4.22. This gain in accuracy was achieved by cancellation of odd powers in time, including the first order term depending on velocity $\mathbf{v}_i$. $\mathbf{v}_i$ is obtained in the Verlet algorithm by subtracting Eq. 4.21 and 4.22. This gives the expression

$$\mathbf{v}_i(t) = \frac{1}{2\delta t}\left[\mathbf{r}_i(t + \delta t) - \mathbf{r}_i(t - \delta t)\right] + O(\delta t^3) \tag{4.24}$$

from which explicit dependence of the forces has been eliminated. The velocity obtained by Eq. 4.24 is the current value at time t, Therefore, the velocity update in the Verlet algorithm is one one step behind the position update. This is not a problem for propagating position because, assuming that the forces are not dependent on velocity, information on $\mathbf{v}_i(t)$ is not needed in Eq. 4.23.

The way velocity is treated in the Verlet algorithm can be inconvenient for the determination of velocity dependent quantities such as kinetic energy. The position and velocity update can be brought in step by a reformulation of the Verlet scheme, called velocity Verlet. The prediction for position is now simply obtained form the Taylor expansion of Eq. 4.21, keeping up to the second order (force) term.

$$\mathbf{r}_i(t + \delta t) = \mathbf{r}_i(t) + \delta t\, \mathbf{v}_i(t) + \frac{\delta t^2}{2m_i}\mathbf{f}_i(t) \tag{4.25}$$

From the advanced position obtained this way we compute the force at time $t + \delta t$

$$\mathbf{f}_i(t+\delta t) = \mathbf{f}_i\left(\{\mathbf{r}_i(t+\delta t)\}\right) = \mathbf{f}_i\left(\left\{\mathbf{r}_i(t) + \delta t\,\mathbf{v}_i(t) + \frac{\delta t^2}{2m_i}\mathbf{f}_i(t)\right\}\right) \tag{4.26}$$

where the curly brackets are an alternative notation for the whole set of coordinates

$$\{\mathbf{r}_i\} = \mathbf{r}^N = (\mathbf{r}_1, \mathbf{r}_2, ..\mathbf{r}_N) \tag{4.27}$$

Substitution of Eq. 4.26 in the Taylor expansion $t \leftarrow t+\delta t$ backward in time using the advanced time $t + \delta t$ as reference

$$\mathbf{r}_i(t) = \mathbf{r}_i(t+\delta t) - \delta t\,\mathbf{v}_i(t+\delta t) + \frac{\delta t^2}{2m_i}\mathbf{f}_i(t+\delta t) \tag{4.28}$$

Added to the forward expansion Eq. 4.25 yields the prediction for velocity

$$\mathbf{v}_i(t+\delta t) = \mathbf{v}_i(t) + \frac{\delta t}{2m_i}[\mathbf{f}_i(t) + \mathbf{f}_i(t+\delta t)] \tag{4.29}$$

which then can be used together with the prediction of position Eq.4.25 in the next step. The (position) Verlet algorithm specified by Eqs. 4.23 and 4.24 and velocity Verlet scheme of Eqs. 4.25 and 4.29 may appear rather dissimilar. They are, however equivalent, producing exactly the same discrete trajectory in time. This can be demonstrated by elimination of velocity. Subtracting from the $t \to t + \delta t$ prediction for position the $t - \delta t \to t$ expansion, we find

$$\begin{aligned}\mathbf{r}_i(t+\delta t) - \mathbf{r}_i(t) &= \mathbf{r}_i(t) - \mathbf{r}_i(t-\delta t) + \\ &\quad \delta t\,[\mathbf{v}_i(t) - \mathbf{v}_i(t-\delta t)] + \frac{\delta t^2}{2m_i}\,[\mathbf{f}_i(t) - \mathbf{f}_i(t-\delta t)]\end{aligned} \tag{4.30}$$

Next the $t - \delta t \to t$ update for velocity

$$\mathbf{v}_i(t) = \mathbf{v}_i(t-\delta t) + \frac{\delta t}{2m_i}\,[\mathbf{f}_i(t-\delta t) + \mathbf{f}_i(t)] \tag{4.31}$$

is inserted in Eq. 4.30 giving

$$\mathbf{r}_i(t+\delta t) - \mathbf{r}_i(t) = \mathbf{r}_i(t) - \mathbf{r}_i(t-\delta t) + \frac{\delta t^2}{m_i}\mathbf{f}_i(t) \tag{4.32}$$

which indeed is identical to the prediction of Eq. 4.23 according to the Verlet scheme without explicit velocities. The Verlet algorithm can be coded up in a few lines. An example is given in the sample code section 4.5.2. This piece of code is also is instructive illustration of reuse and updating of variables.

4.2 Introducing temperature

4.2.1 Time averages

The sequence of positions $\mathbf{r}^N(t_m)$ and velocities $\mathbf{v}^N(t_m)$ at the discrete time points $t_m = m\delta t, m = 1, \cdots M$ generated by a successful MD run represents a continuous trajectory $\mathbf{r}^N(t)$

of the system of duration $\Delta t = M\delta t$ with starting point $t = 0$ and end point $t = \Delta t$. We can use this discrete trajectory to visualize the motion of the particles on a graphics workstation, but in the end we always want to compute a time average of some observable $\mathcal{A}$. The total energy $\mathcal{K} + \mathcal{V}$ of Eq. 4.14 is an example of an observable. Written as a function of position $\mathbf{r}^N$ and momentum $\mathbf{p}^N$, observables are usually called phase functions, for which we will use the notation $\mathcal{A}\left(\mathbf{r}^N, \mathbf{p}^N\right)$.

Evaluated along a given trajectory $\mathbf{r}^N(t)$, they yield an ordinary function $A(t)$ of time

$$A(t) \equiv \mathcal{A}\left(\mathbf{r}^N(t), \mathbf{p}^N(t)\right) \tag{4.33}$$

which are, of course, different for different trajectories. $A(t)$ is a proper function of time and, accordingly, can be can be differentiated with respect to time giving

$$\frac{dA}{dt} = \frac{d}{dt}\mathcal{A}\left(\mathbf{r}^N(t), \mathbf{p}^N(t)\right) = \sum_{j=1}^{N}\left[\dot{\mathbf{r}}_j \cdot \frac{\partial \mathcal{A}}{\partial \mathbf{r}_j} + \dot{\mathbf{p}}_j \cdot \frac{\partial \mathcal{A}}{\partial \mathbf{p}_j}\right] \tag{4.34}$$

A similar application of the chain rule to obtain a (total) time derivative was already applied in the proof of the conservation of energy (Eq. 4.16). $A(t)$ can also be averaged, i.e integrated over time. Denoting the time average of the phase function over the continuous trajectory $\mathbf{r}^N(t)$ of length Δt by $\overline{A}_{\Delta t}$ we can write

$$\overline{A}_{\Delta t} = \frac{1}{\Delta t}\int_0^{\Delta t} dt A(t) = \frac{1}{\Delta t}\int_0^{\Delta t} dt \mathcal{A}\left(\mathbf{r}^N(t), \mathbf{p}^N(t)\right) \tag{4.35}$$

Since the time step in MD is supposed to be smaller than the fastest motion in the system the average of the discrete points of the MD trajectory gives us a very good approximation to $\overline{A}_{\Delta t}$.

$$\overline{A}_{\Delta t} \cong \frac{1}{M}\sum_{m=1}^{M} A(t_m) \tag{4.36}$$

4.2.2 ∗Ensemble averages

Time averages also provide the connection to statistical mechanics through the ergodic principle. This principle states that time averages of ergodic systems, in the limit of trajectories of infinite length Δt, can be replaced by ensemble averages (see Fig. 4.1 and the part II course on statistical mechanics). Since the MD algorithms discussed so far (ideally) produce a trajectory at constant energy, the appropriate ensemble for MD is the microcanonical ensemble.

$$\lim_{\Delta t\to\infty}\overline{A}_{\Delta t} = \int d\mathbf{r}^N d\mathbf{p}^N \rho_{NVE}\left(\mathbf{r}^N, \mathbf{p}^N\right)\mathcal{A}\left(\mathbf{r}^N, \mathbf{p}^N\right) \equiv \langle A\rangle_{NVE} \tag{4.37}$$

where ρ_{NVE} is given by a Dirac delta function in the total energy, restricting the manifold of accessible phase points $\mathbf{r}^N, \mathbf{p}^N$ to a hypersurface of constant energy E only.

$$\begin{aligned}\rho_{NVE}\left(\mathbf{r}^N, \mathbf{p}^N\right) &= \frac{f(N)}{\Omega_N}\delta\left[\mathcal{H}\left(\mathbf{r}^N, \mathbf{p}^N\right) - E\right] \\ \Omega_N &= f(N)\int d\mathbf{r}^N d\mathbf{p}^N \delta\left[\mathcal{H}(\mathbf{r}^N, \mathbf{p}^N) - E\right]\end{aligned} \tag{4.38}$$

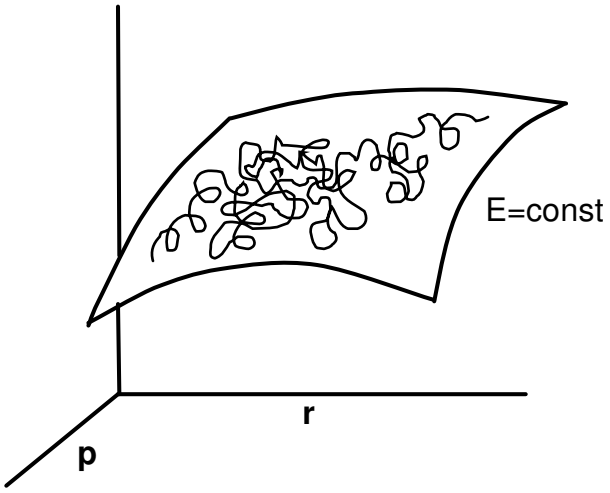


Figure 4.1: Schematic representation of a constant energy surface in phase space with chaotic trajectory. Provided the system is ergodic, the trajectory will eventually fill up the entire constant energy plane, i.e. visit every point on the surface arbitrarily closely, allowing us to replace the time average over the trajectory by an geometric average over the hypersurface.

$\mathcal{H}$ is the phase function defined in Eq. 4.14, giving the total energy of the system. $f(N)$ is some function of the number of particles, which can be omitted if we are only interested in averages over the ensemble distribution ρ_{NVE}. This factorbecomes crucial if we want to give the normalization factor Ω_N a thermodynamical interpretation (namely entropy, see below). The ergodic property which is the basis of the equivalence of time and ensemble averages (Eq. 4.37) is an assumption valid for stable many-particle systems. However, there systems for which this condition is not satisfied, such as glasses.

Condensed matter systems are hardly ever isolated. The least they do is exchanging energy with their environment. In the part II statistical mechanics lectures and any textbook on the subject, it is shown that states of such a system, in equilibrium with a thermal reservoir of temperature T, are distributed according to the canonical ensemble.

$$\begin{aligned} \rho_{NVT}\left(\mathbf{r}^N, \mathbf{p}^N\right) &= \frac{f(N)}{Q_N} \exp\left[-\frac{\mathcal{H}\left(\mathbf{r}^N, \mathbf{p}^N\right)}{k_B T}\right] \\ Q_N\left(V, T\right) &= f(N) \int d\mathbf{r}^N d\mathbf{p}^N \exp\left[-\frac{\mathcal{H}\left(\mathbf{r}^N, \mathbf{p}^N\right)}{k_B T}\right] \end{aligned} \tag{4.39}$$

Canonical expectation values are exponentially weighted averages over all points in phase space

$$\begin{aligned} \langle A \rangle_{NVT} &= \int d\mathbf{r}^N d\mathbf{p}^N \rho_{NVT}\left(\mathbf{r}^N, \mathbf{p}^N\right) \mathcal{A}\left(\mathbf{r}^N, \mathbf{p}^N\right) \\ &= \frac{f(N)}{Q_N} \int d\mathbf{r}^N d\mathbf{p}^N \mathcal{A}\left(\mathbf{r}^N, \mathbf{p}^N\right) \exp\left[-\beta \mathcal{H}\left(\mathbf{r}^N, \mathbf{p}^N\right)\right] \end{aligned} \tag{4.40}$$

where as usual $\beta = 1/k_B T$. The canonical ensemble also provides an easy route to obtain the expression for the factor $f(N)$ by taking the classical limit of the quantum canonical ensemble. Again we refer to the Part II lectures for details of the derivation. If all N particles are identical

(of the same species) the result is

$$f(N) = \left(h^{3N} N!\right)^{-1} \tag{4.41}$$

where h is Planck's constant. With this factor $f(N)$ Q_N included of Eq. 4.39 and Ω_N of Eq. 4.38 are known as respectively the canonical and microcanonical partition function. Their interpretation is suggested by considering the dimension of h, which is that of position $\times$ momentum. $f(N)$ in Eq. 4.41 is therefore a (very small) reciprocal phase space volume which makes the canonical partition function of Eq. 4.39 a dimensionless quantity i.e. a real number. Planck's constant acts, therefore, as an absolute measure of phase space metric and Q_N is interpreted as the effective number of accessible states at temperature T. The $N!$ takes account of the indistinguishability of the particles. It can be viewed as correcting for overcounting in the classical ensemble where permuting the position and momentum of a pair of particles would lead to a different, but equivalent, state (point) $\mathbf{r}^N, \mathbf{p}^N$ in phase space. The factor Ω_N of Eq. 4.38 has a similar interpretation in terms of an accessible number of states, except that microcanonical motion is restricted to a hyper surface in phase space (i.e a manifold of dimension 6N-1). A mathematical more correct way of thinking about the microcanonical partition function is that for given infinitesimal energy dE, the quantity $\Omega_N dE$ gives the effective number of states contained in the volume between hypersurfaces with energy E and $E + dE$.

Ω_N and Q_N can be related to two very important thermodynamic quantities, namely Ω_N to the Boltzmann entropy S

$$S = k_B \ln \Omega_N \tag{4.42}$$

and Q_N to the Helmholtz free energy A.

$$A = -k_B T \ln Q_N \tag{4.43}$$

where k_B is Boltzmann's constant. Eqs. 4.42 and 4.43 are the central relations linking statistical mechanics to thermodynamics. The factor $f(N)$ played a crucial role in this identification. It is helpful not to forget that the founding fathers of statistical mechanics arrived at these results without the help of quantum mechanics. Arguments concerning the additivity of entropy of mixing and similar considerations led them to postulate the form of the N dependence. It was, of course, not possible to guess the precise value of the effective volume of the microscopic phase element h^{3N}.

Kinetic energy is a rather trivial quantity in (classical) statistical thermodynamics. The average per particle is, independently of interaction potential or mass, always equal to $3/2 k_B T$ (equipartition). The basic quantity of interest is the probability distribution $P_N\left(\mathbf{r}^N\right)$ for the configuration $\mathbf{r}^N$ of the system obtained by integrating over momenta in Eq. 4.39. Omitting the normalization factor $f(N)$ we can write P_N as

$$\begin{aligned} P_N\left(\mathbf{r}^N\right) &= \frac{1}{Z_N} \exp\left[-\beta \mathcal{V}\left(\mathbf{r}^N\right)\right] \\ Z_N &= \int^V d\mathbf{r}^N \exp\left[-\beta \mathcal{V}\left(\mathbf{r}^N\right)\right] \end{aligned} \tag{4.44}$$

The configurational partition function Z_N in Eq. 4.44, is the integral of the Boltzmann exponent $\exp\left[-\beta \mathcal{V}\left(\mathbf{r}^N\right)\right]$ over all configuration space. The extension of configuration space is defined by the volume V in which the particles are contained, setting boundaries for the integration over spatial coordinates. This restriction, which was implicit in Eqs. 4.38 and 4.39, has been

made explicit in Eq. 4.44. Z_N is related to the canonical partition function Q_N and the free energy by

$$\exp\left[-A/k_BT\right] = Q_N = \left(N!\Lambda^{3N}\right)^{-1} Z_N \tag{4.45}$$

where Λ is the thermal wavelength

$$\Lambda = \frac{h}{\sqrt{2\pi m k_B T}} \tag{4.46}$$

The factor Λ^{3N} is a temperature dependent volume element in configuration space. The deeper significance of the thermal wavelength Λ is that it provides a criterion for the approach to the classical limit. Quantum effects can be ignored in equilibrium statistics if Λ is smaller than any characteristic length in the system. Again we refer to the course on Statistical Mechanics for further explanation.

4.2.3 Temperature in MD and how to control it

Temperature was introduced in section 4.2.2 as a parameter in the exponent of the canonical ensemble distribution function Eq. 4.39. Via the fundamental Eq. 4.43 this statistical temperature could be identified with the empirical temperature of classical thermodynamics. It is not immediately obvious, however, how to use these concepts to define and measure temperature in a MD simulation. For this we have to return to the microcanonical ensemble and find an observable (phase function) $\mathcal{T}$ for which the microcanonical expectation value is a simple function of temperature, preferably linear. This temperature could then also be measured by determining the time average of the phase function $\mathcal{T}$ over a sufficiently long period, because Eq. 4.37 allows us to equate the time average and microcanonical ensemble average. In fact, this is very much how real thermometers work. For classical systems there is such a phase function, namely kinetic energy. The canonical average of kinetic energy is particularly easy to compute

$$\left\langle \sum_{i=1}^{N} \frac{\mathbf{p}_i^2}{2m_i} \right\rangle_{NVT} = \frac{3}{2} N k_B T \tag{4.47}$$

The microcanonical average $\langle \cdots \rangle_{NVE}$ of Eq. 4.37 and canonical average of Eq. 4.40 of a quantity are in principle not identical. In the statistical mechanics course it is shown that for properties such as kinetic energy, the difference is one order less in system size N. This implies that the fractional difference vanishes in the thermodynamic limit of very large N. The microcanonical average of the kinetic energy of a many particle system, therefore, will also approach $\frac{3}{2}Nk_BT$. Hence, we can define an instantaneous or kinetic temperature function

$$\mathcal{T} = \frac{1}{3k_BN} \sum_{i=1}^{N} m_i \mathbf{v}_i^2 \tag{4.48}$$

which, averaged over a MD run gives us the temperature of the system (see Eq. 4.36)

$$T = \frac{1}{M} \sum_{m=1}^{M} T(t_m) \tag{4.49}$$

The formal way we have introduced kinetic temperature, is clearly somewhat heavy and redundant for such a simple property. However, for other quantities, such as pressure, the relation between the corresponding mechanical observable and their thermodynamic counterpart is less straightforward.

After having found a method of measuring temperature in MD, the next problem is how to impose a specified temperature on the system and control it during a simulation. Several approaches for temperature control in MD have been developed, some more sophisticated and rigorous than others. For the purpose of getting started, the most suitable algorithm is the simplest, and also the most robust, namely temperature scaling. The idea is to scale all particle velocities by a factor determined from the ratio of the instantaneous kinetic temperature and the desired temperature. Suppose the current (instantaneous) temperature $\mathcal{T}$ of Eq. 4.48 is considerably different from our desired target temperature and we want to adjust it to our target value T. Rescaling all of the current velocities $\mathbf{v}_i$ according to

$$\mathbf{v}_i' = \sqrt{\frac{T}{\mathcal{T}}}\,\mathbf{v}_i \tag{4.50}$$

will do the job because after squaring and adding the new velocities according to Eq. 4.48 we have a new temperature

$$\mathcal{T}' = \frac{1}{3k_BN}\sum_i^N m_i\left(\mathbf{v}_i'\right)^2 = \frac{1}{3k_BN}\sum_i^N m_i\frac{T}{\mathcal{T}}\mathbf{v}_i^2 = T \tag{4.51}$$

An example of the simple code for rescaling velocities in combination with the Verlet Algorithm can be found in section 4.5.3.

Under equilibrium conditions, velocities are distributed according to a gaussian leading to the famous Maxwell Boltzmann distributions. The probability functions for each of the three Cartesian components of velocity of every particle i is strictly Gaussian,

$$P\left(v_{x,i}\right) = \sqrt{\frac{m_i}{2\pi k_BT}}\exp\left[-\frac{m_iv_{x,i}^2}{2k_BT}\right] \tag{4.52}$$

and the same for $v_{y,i}, v_{z,i}$. Temperature scaling (Eq. 4.50) only controls the width of the velocity distribution, it will not change a non-equilibrium distribution into a Gaussian. Due to the chaotic motion of the particles the velocity distribution should eventually converge to a Gaussian, also in our model system. However, it can take awhile before equilibrium has been established. We can accelerate the equilibration process by interfering with the dynamics more strongly and randomize the velocities by a sampling from a gaussian distribution. This would also be the best way to initialize the velocities when we start a simulation from a given set of positions. Section 4.5.3 outlines a simple scheme for turning a uniform random number generator as available on most computers (see section 4.5.1) into a procedure for sampling from an (approximate) Gaussian distribution.

4.3 Force computation

4.3.1 Truncation of short range interactions

Having introduced the basic procedures for propgating the particle coordinated given the forces, we now turn to the task of computing the forces. The interaction model we will use is the pairwise additive potential which has become the prototype for MD, namely the 12-6 Lennard-Jones

potential. The pair potential $v(r)$ (see Eq. 4.5) defining this model is usually written in the form

$$v(r) = 4\epsilon \left(\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right) \tag{4.53}$$

in which the interaction strength ϵ and interaction range σ have a convenient interpretation: $v(r) = 0$ at $r = \sigma$, repulsive for $r < \sigma$ and attractive for $r > \sigma$ with a minimum of $v(r_0) = -\epsilon$ at $r_0 = 2^{1/6}\sigma \approx 1.12\sigma$. For large distances the potential $v(r)$ approaches zero. Good 12-6 parameters for liquid argon are $\epsilon/k_B = 120K$ and $\sigma = 3.4$Å.

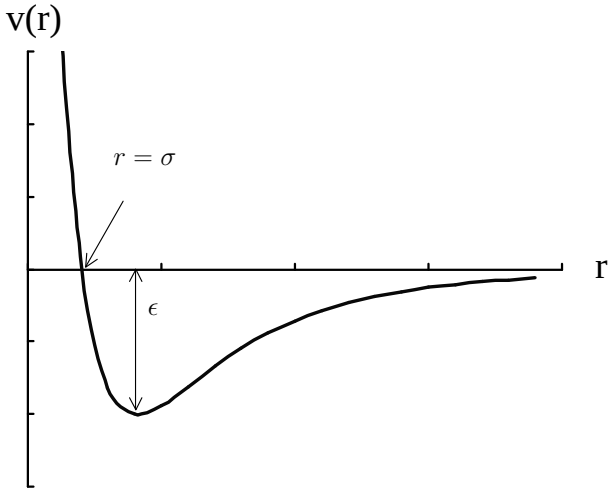


Figure 4.2: Lennard-Jones 12-6 potetntial of Eq. 4.53

At $r = 3\sigma$, $v(r) \approx -0.005\epsilon$, i.e. less than a percent of the value at the minimum. Therefore, beyond this radius, or even already at shorter distances, the contribution to energy and forces can be neglected, which saves computer time. The actual potential that will be used in the force calculation of Eq. 4.8 is the truncated function:

$$v_c(r) = \begin{cases} v(r) & r \leq r_c \\ 0 & r > r_c \end{cases} \tag{4.54}$$

where r_c is called the cutoff radius. The code for the force calculation for the pair potential of section*

As a conclusion of this section, a brief comment on the ultimate short range model, hard spheres. This interaction is described by the pair potential

$$v(r) = \begin{cases} +\infty & r \leq \sigma \\ 0 & r > \sigma \end{cases} \tag{4.55}$$

where the interparticle distance $r = r_{ij} = [(x_i - _j)^2 + (y_i - y_j)^2]^{1/2}$ in the 2D system (hard disks) and $r = r_{ij} = [(x_i - _j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2]^{1/2}$ in the 3D system (hard spheres). There is no attraction, only infinite repulsion as soon as the hard cores with diameter σ overlap. This model is obviously not capable of accounting for the cohesion in liquids: Hard sphere fluids, therefore, show no vapour-liquid transition. However, it gives a very good first approximation for the local structure in liquids and the disorder distinguishing a liquid from a solid. These effects can be explained, at least qualitatively, by excluded volume only (see e.g. the book of

Chandler). This simple model is therefore very popular in Monte Carlo. In fact, MC simulations of hard sphere fluids have made vital contributions to our understanding of liquids. For MD simulation, however, the hard sphere potential rather cumbersome because of the singularities in the derivatives (forces). The standard model for MD studies of liquids is the 12-6 Lennard-Jones potential of Eq. 4.53.

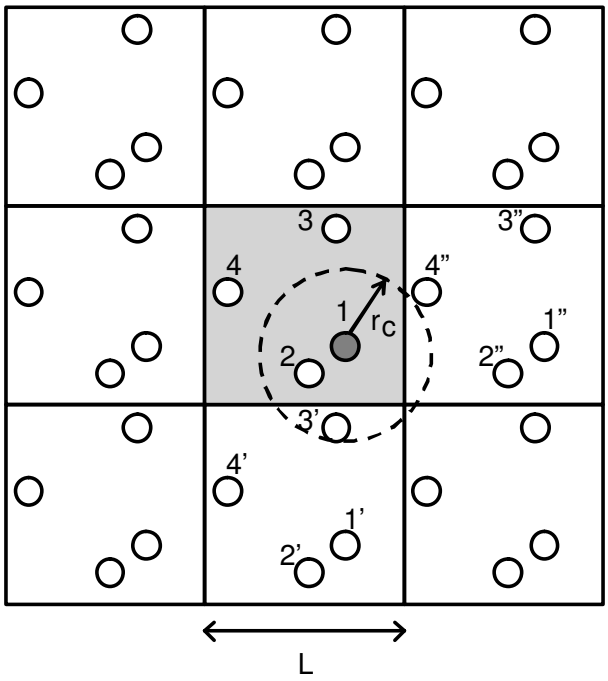


Figure 4.3: Periodic boundary conditions with spherical cutoff and minimum image convention. Particles 2 and the image 3' of particle 3 are inside cutoff sphere around particle 1. All images of particle 4 are outside.

4.3.2 Periodic boundary conditions

It is relatively straightforward to implement the computation of pair forces for a finite set of N particles which can be located anywhere in space. Code 4.27 of the appendix gives an example how this can be coded up. These boundary conditions correspond to a cluster of atoms in vacuum. In order to describe liquids with uniform (average) density we can either take a very big cluster and hope that in the interior of the cluster surface effects can be neglected, or use periodic boundary conditions. Periodic boundary conditions replicate a MD cell with the shape of a parallelepiped, and its contents, all over space mimicking the homogeneous state of a liquid or solid. Of course, the periodic nature will introduce certain errors,called finite size effects, which can be small or rather serious depending in the nature of the system. If the MD box is spanned by the three vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ the images are displaced by multiples $l\mathbf{a} + m\mathbf{b} + n\mathbf{c}$ of these basis vectors, where l, m, n are integers (positive and negative). The potential energy of the particles in the central cell, corresponding to $(l, m, n) = (0, 0, 0)$, is now a sum of the

interactions over all cells.

$$\mathcal{V}(\mathbf{r}^N) = \frac{1}{2}\sum_i^N v_i(\mathbf{r}^N)$$

$$v_i(\mathbf{r}^N) = \sum_{l,m,n=-\infty}^{+\infty} {\sum_{j=1}^{N}}' v\left(|\mathbf{r}_j + l\mathbf{a} + m\mathbf{b} + n\mathbf{c} - \mathbf{r}_i|\right) \tag{4.56}$$

where the $'$ indicates a condition on the summation excluding $j = i$ for $l, m, n = 0$ (self interaction in the central cell). Note that linear momentum is still a constant of motion in such a set of infinitely replicated systems. The conservation of angular momentum, however is lost, as a result of the reduction of rotational symmetry from spherical to cubic.

For short range interactions such as the 12-6 interaction of Eq. 4.53 it is possible to make the size of the system sufficiently large that the contributions of all images, except the nearest, can be disregarded, because they are too far away. The nearest image can be in the same (i.e. central) cell but also in one of the neighboring cells (see Fig.4.3). This approximation is known under the name minimum image approximation. Again a glance at an actual code implementation of the minimum approximation can be very instructive. An example is given in section 4.5.4.

4.4 MD in practice

4.4.1 System size

Given an interaction model (the Lennard-Jones 12-6 pair potential in most of the examples treated in this course), there are a number of choices to make in setting up a molecular dynamics simulation. First of all there is the size of the model system. The thermodynamic state of a (pure) liquid is specified by only two parameters, temperature T and pressure P, or alternatively temperature T and number density $\rho = N/V$ (the number of particles N per volume V). The number of particles in the model system is related to the density and length L of the side of the cubic periodic cell as $N = \rho L^3$. The size of a model for a liquid of specified density ρ is therefore completely determined by the choice of L. While in the early days of the development of the MD method, the number of Lennard Jones atoms was typically in the order of a 100, the rapid progress in the performance of hardware, has continuously increased this number, and simulation of systems consisting of a 10^5 Lennard-Jones atoms are quite common in these days.

System size in MD is in practice a compromise between the length scale of the problem of interest and the minimum duration of a run required for proper statistical sampling. Clearly, if we are interested in, for example, the onset of freezing the system must be much larger than needed for a study of the structure of the radial distribution of a stable liquid. Similarly, computation of transport properties, such a diffusion coefficients, will require much longer runs than estimation of internal energy.

4.4.2 Choosing the time step

The time step δt is a key parameter in a MD simulation. A longer time step is clearly more economical, because the computer time it takes to follow the system of a certain stretch of real time (say 100 ps) will be less. However, the errors in the numerical integration rapidly

accumulate with the length of the time step, and usually there is a fairly well defined maximum time step beyond which the propagation breaks down. The best indication of this break down is drift in the total energy which should be rigorously conserved according to Newtons equation of motion (section 4.1.2). This is illustrated in Figure 4.4 for a fluid of Lennard-Jones particles. Time t^* is measured in reduced units (see also section 4.5.5).

Figure 4.4: Energy conservation for a Lennard-Jones fluid of 108 atoms in cubic box using the Verlet algorithm. The left panel shows the potential energy, kinetic energy and the sum of these two over 5000 steps for an optimized value of the time step ($\delta t^* = 0.005$ in reduced units, see Eq. 4.57). The total energy is effectively conserved. The right panel shows what happens if the time step is increased to $\delta t^* = 0.015$ again for 5000 steps. The total energy now exhibits a small drift indicating that this time step is too long.

$$t^* = t\left(\frac{\epsilon}{m\sigma^2}\right)^{1/2} \tag{4.57}$$

The length of a time step depends on the type of propagation algorithm. The more accurate the propagation, the longer the time step can be. For example, the maximum time step for integrating the equation of motion of an harmonic oscillator using a Verlet algorithm is in the order of 1/20 of the period of oscillation (see also section 4.5.2). More advanced propagation algorithms can tolerate much longer time steps, in particular for harmonic potentials. However, the time step also varies with the magnitude of the forces, and therefore the steepness of the potential. Steeper potentials require shorter time steps. What matters, of course, is the region of the potentials that is actually accessed in the simulation. For example, for repulsive force between two particles interacting via a Lennard Jones 12-6 Potential becomes arbitrarily large the closer the particles approach each other.

These and other practical and technical consideratons encountered in a MD simulation are further clarified in section 4.5.5 of the appendix, where the full code for a MD simulation of liquid argon is outlined in some detail.

4.4.3 ∗Why use the Verlet algorithm?

While there are algorithms with better short time accuracy than the Verlet algorithm, the overwhelming majority of condensed matter MD simulation is based on just this algorithm. There are a number of reasons for its popularity

i The Verlet algorithm is simple and efficient depending only on forces. No higher energy derivatives are needed. This is important because the force evaluation is the most CPU time consuming operation in MD simulation of interacting many particle systems. Computation of higher derivatives of the potential energy function w.r.t to the particle coordinates would be too demanding in terms of computational costs. Moreover, while algorithms using force derivatives are more accurate, and therefore can be integrated with a longer time step, the gain in time step is relatively minor. Because of the chaotic nature of the motion in many-particles systems the particles rapidly deviate from the "true" trajectories. In fact this deviation is exponential in time (this type of strong molecular chaos is ultimately the justification for the application of methods of statistical mechanics.)

ii Even though only using forces, the Verlet algorithm is correct up to and including third order (δt^3).

iii The Verlet algorithm is explicitly time reversible and, even though a discretized trajectory relatively quickly diverges substantially from the true trajectory (typically in 100 steps) the energy is conserved over much longer times (see Fig. 4.4). Moreover, the Verlet algorithm rigorously conserves the normalization of an ensemble probability distribution of points in phase space (see FS). These rather formal qualities contribute to the superior long time stability of the Verlet algorithm. For example, as discussed in section 4.2.2, energy was the defining quantity for the microcanonical ensemble, since, for chaotic systems, there are no constraints on the regions trajectories can reach in in phase space other than that they are confined to an hypersurface of constant energy. Energy conservation, together with norm conservation are therefore necessary conditions for thermodynamic stability, and ultimately for a proper definition of temperature. Long time stability is particularly important for the simulation of liquids which are stabilized by finite temperature dynamical fluctuations.

4.5 Appendix 1: Code samples

4.5.1 ∗Pseudo code

Assignment and conditional statements

In order to outline some of the essential points of the translation of numerical schemes such as the Verlet algorithm of section 4.1.3 into computer code we introduce a kind of pseudo programming language. This pseudo language is patterned after fortran and can be viewed as a "sloppy" version of it meant for humans rather than machines. The basic statement in programming languages is an assignment which asks the computer to perform certain operations on certain data and store the result. We will take as an example the instruction to increase the value of an integer n by 1. We will write this most basic of operations as

$n := n + 1$

Code 4.1: Assignment statement

It is convenient to add a comment in the text of our pseudo code explaining the operation at hand. We will place the text of the comment in a special type of brackets, $\backslash * \cdots * \backslash$, which in actual programming languages tells the compiler to ignore this line.

$n := n + 1$ $\backslash *$ increase n by 1 $* \backslash$

Code 4.2: Adding comments

A further crucial bit of code is a conditional statement as in the following example

if $(n > 0)$ *then*
    $n := n - 1$
else
    $n := n + 1$
endif

Code 4.3: *if* statement

The effect of the *if* statement in 4.3 is that n is decreased by 1 when positive and increased by 1 otherwise. In order to iterate an operation, we need a command instructing the machine to repeat a this operation a certain amount of times. This is in fortran achieved with a *do* statement which will be written here as

```
m := 100                                        \* initialize m and n *\
n := 0
do i := 1, m                                      \* repeat m times *\
      if (n > 0) then
           n := n − 1
      else
           n := n + 1
      endif
enddo
```

Code 4.4: *do* loop

The result will be a string of 1's alternating with 0's with a total length of 100.

Use of arrays
The way the code 4.4 is set up now, the next value of n will overwrite the previous one and the data will be lost unless we write to output or convert n into a array $n(i)$. The array $n(i)$ is a indexed set of variables, n being its name and i the (integer) index. In the context of our simple pseudo code we can dispense with for a computer crucial technicalities of type and length declaration. Replacing the scalar n in the *do* loop by an array with the appropriate indexing we obtain

```
m := 100                                        \* initialize m and n *\
n(1) := 0
do i := 1, m − 1
      if (n(i) > 0) then
           n(i + 1) := n(i) − 1                  \* determine next value of n *\
      else
           n(i + 1) := n(i) + 1
      endif
enddo
```

Code 4.5: Example use of arrays

Note that in order to generate exactly m numbers, we initialize the first value and obtain the the rest by performing $m - 1$ iterations. The index of a do loop (i in Code 4.4 and Code 4.5)) is also available as a variable in assignment or other statements provided these instructions are *inside* the *do* loop. A very simple example making use of this feature of *do* loops (or whatever they are called in a specific language) is the code below which puts 100 particles on a line at equal distance

```
natom := 100                                    \* number of particles *\
dx := 1                                           \* particle spacing *\
do i := 1, natom
      x(i) := i ∗ dx
```

enddo

Code 4.6: Putting particles on a line

The $*$ in Code 4.6 is the usual symbol for multiplication in computer languages. For completeness, we list below the syntax for this and other arithmetic operations

$a * b = ab$	multiplication
$a/b = \frac{a}{b}$	division
$a * *b = a^b$	exponentiation

Code 4.7: Some arithmetic operations

Subroutines

Suppose later on in the program we want to construct another series of points on a line. Instead of repeating the lines of Code 4.6 we can turn them into a subprogram or procedure which can be executed whenever needed. First, we have to define the procedure by placing the relevant code between a *subroutine* and *end* statement.

```
subroutine line (natom, x)
dx := 1                                          \* particle spacing *\
do i := 1, natom
      x(i) := i * dx
enddo
end
```

Code 4.8: Defining a subroutine

The arguments attached to the *subroutine* definition statement are the parameters required to perform its task, in this case the number of atoms *natom*, and the variables x in which, on completion, the results are stored and made available for further use. The procedure is executed by calling. For example, preparing a particle array and writing the positions to an output file using a *print* statement can be achieved by

```
read natom
call line (natom, x)
print x                                                          (C8)
```

Code 4.9: Reading input, calling a subroutine and writing output

Code 4.9 also illustrates our representation of a further basic function, namely reading in from input. This is performed by the *read* statement, which reads the value of *natom* either from the terminal or some file stored on disk. We can identify the little program of Code 4.9 to the computer, and to humans, by prefacing it with a *program* statement. giving its name, and closing with an *end* statement.

```
program make_line
read natom, line
  \* choose one of two options depending on the value of the logical variable line *\
if (line) then
      call line (natom, x)                       \* in case line = .true. *\
```

```
else
      do i = 1, natom                                  \* in case line = .false. *\
           x(i) := 0
      enddo
endif
print x
end
```

Code 4.10: A complete program in simplified code

In order to make the program more flexible, we want to have the option of executing different tasks. This is the purpose of reading in the control parameter *line* in Code 4.10: *line* is a logical (binary) variable which can either have the value ".true." or ".false.". As such it can be directly used as the conditional argument of *if* statements.

Intrinsic functions
We will also frequently use a modification of the *subroutine* procedure called a *function*. The difference is that the function name is itself a variable with a value assigned to it by the procedure. The simplest examples are the intrinsic functions supplied by the system. Below is a list of some of the the most common functions

$y := sqrt(x)$	square root $y = \sqrt{x}$
$y := exp(x)$	exponent $y = e^x$
$y := log(x)$	logarithm $y = \ln(x)$
$y := abs(x)$	absolute value $y = \|a\|$
$n := int(x)$	truncation (n is largest integer with $\|n\| \leq \|x\|$)
$n := nint(x)$	nearest integer
$z := max(x, y)$	Choosing largest value
$z := min(x, y)$	Choosing smallest value
$z := mod(x, y)$	remainder $z = x - int(x/y) * y$

Code 4.11: Common arithmetic and mathematical functions

It is also possible to define a user *function* similar to the definition of the *subroutine* procedure of Code 4.8. As an example we will make our own nearest integer function, which should give the same result as the intrinsic *nint* function of Code 4.11.

```
function my_nint (x)
it (x ≥ 0) then
      my_nint = int (x + 0.5)
else
      my_nint = int (x − 0.5)
endif
```

Code 4.12: Definition of function for the determination of the nearest integer

Exercise 1: *Examine the effect of the function* $y = x - l * nint(x/l)$ *on the series* $x_n = z + nl/2$ *where* $0 < z \leq l/2$ *and* n *is an integer with the values* $-3, -2, -1, 0, 1, 2, 3$. *This property of the nint function will be utilized later in the code for the application of periodic boundary conditions.*

Finally, to give a more complicated example using several of the syntactic elements introduced above, we sketch the pseudo code for a three dimensional generalization of the procedure of Code 4.8 placing the particles on a cubic grid. The Cartesian components of the coordinate vector $\mathbf{r}_i$ are stored in the array elements $x(i), y(i), z(i)$. l is the side of the cube. This parameter and *natom* are read from input.

```
    subroutine init_r (natom, l, x, y, z)
    hl = l/2
    nl = int (natom**(1/3))                          \* integer truncation of N^(1/3) *\
    if (nl**3 < natom)  nl := nl + 1                 \* increase by 1 if N ≠ n^3 *\
    dl = l/nl
    n = 0
    do i := 0, nl - 1
        do j := 0, nl - 1
            do k := 0, nl - 1
                n := n + 1
                if (n > natom) goto 10               \* stop if all particles are done *\
                x(n) := i * dl - hl
                y(n) := j * dl - hl
                z(n) := k * dl - hl
            enddo
        enddo
    enddo
10  end
```

Code 4.13: Placing particles on a cubic grid

In first two statements, the number of lattice points nl along a side of the cube is determined. The function $int(x)$ truncates a (positive) real number x to the largest integer smaller than x. nl must be large enough to fit all $N = natom$ points in the cube. Therefore, for all *natom*, except third powers of integers $N = n^3$, nl as estimated in the first line, will come out too small. Hence, the increase by 1 in the next line. Now, of course, nl will be too large for most *natom* and there will be unoccupied sites in the cube. Therefore, we must jump out of the three nested *do* loops covering all cubic lattice points when all *natom* particles have been assigned to a lattice point. This is achieved by the conditional *go to* statement, which transfers execution directly the statement 10, which in this case is the end of the procedure.

Exercise 2: *Code 4.13 will be applied repeatedly to set up a system of particles in a cubic periodic cell which serves as the starting configuration for the simulation of a liquid. The simple cubic (sc) lattice generated by Code 4.13 is of course far removed from a typical configuration of a liquid. Better would be a face centered cubic (fcc) lattice which is locally more similar to the closed packed coordination encountered in (simple) liquids. Can you generalize Code 4.13 for the construction of fcc lattices?*

Random number generators

Most compilers provide an uniform random number generator in the form of an intrinsic function. Random number generators are needed in MD for initialization of velocities (and sometimes positions). A uniform random number procedure generates sequences of real numbers

in the interval $[0, 1)$ with equal probability and as little correlation as possible. The syntax of this call, while available in the default numerical system library, is not standard and may differ depending on what make of computer you have. We will follow the conventions used on workstations (Compac, Sun) and, in order to be general, we will construct a little shell subroutine around it, so only this routine will require platform dependent adjustments.

```
subroutine my_ran (n, rndm, iseed)
if (iseed ≤ 0)    iseed := 87654321          \* initialize seed if necessary *\
                                                \* Dec and Sun format *\
do i := 1, n
     rndm(i) := ran(iseed)
enddo
end
```

Code 4.14: Generate n uniform random numbers

The presence of the variable *iseed* is vital: It is a large positive (usually) integer number which is the actual variable iterated by the random number generator. The variable x is only a derived quantity. This means that the procedure *ran* in Code 4.14 transforms the value of *iseed* supplied to it as an argument and returns the updated seed in the same variable *iseed* which must be passed on as input for the next call to *ran*. The important point to keep in mind when dealing with subroutines like *ran* is that the generation of the random numbers is entirely deterministic: The same value of *iseed* as input will always produce the same result. This is why random number generators on computers are sometimes called pseudo random number generators. The results, however, are unpredictable and show little statistical correlation. They can, therefore, be used as an approximation to a series of random numbers. The *if* statement in Code 4.14 is just a safety against uninitialized seeds. Again, when we leave setting the seed to this line, the series of random numbers produced will be always identical.

4.5.2 ∗Coding up Verlet

Verlet for the harmonic oscillator

The simplified computer language introduced in section 4.5.1 gives us the tools to sketch how the Verlet algorithm of Eqs. 4.23 and 4.24 can be implemented in a computer code. For the specification of the position vectors $\mathbf{r}_i$ we could use three separate component arrays $x(i), y(i), z(i)$, similar to Code 4.13. An alternative would be to store all coordinates in a $N \times 3$ array $r(i, k)$, where the index $i = 1, \cdots N$ counts the particles and $k = 1, 2, 3$ the components x, y, z. In order to keep the number of indices to a minimum we will opt for simply writing down instructions for each component separately, as we did in Code 4.13.

The subroutine *init_r* of Code 4.13 can directly be utilized to set up the initial values of the particle coordinates for the Verlet algorithm. We also have to construct a set of velocities because initial conditions in Newtonian dynamics require specification of both positions and velocities. As explained in section 4.1.3, a peculiarity of the original Verlet algorithm Eq 4.25 is that velocities are secondary quantities not appearing in the prediction for position coordinates. Instead, the position at the previous step $t - \delta t$ is used. Therefore, we must introduce a second set of coordinate arrays $xm(i), ym(i), zm(i)$ representing the positions at $t - \delta t$. At the start of the MD iteration, these must have already a value, hence we need to initialize these variables as well. For the moment we will defer discussion of proper initialization of velocity after we

have introduced temperature in MD, and will simply give add to the $t = 0$ coordinates as constructed by the position coordinate initialization routine Code 4.13 a small offset consistent with a uniform velocity $v0$. We will make a procedure that does this for each component separately.

```
subroutine init_v (natom, dt, v0, x, xm)
dx := v0 * dt                          \* uniform offset for xm, dt is the time step *\
do i := 1, natom
     xm(i) := x(i) - dx
enddo
end
```

Code 4.15: Uniform initialization of velocity component for Verlet algorithm

After initialization of the coordinates, we can proceed with the iteration according to the forces acting on the particles. We will assume a very simple force model, namely an isotropic harmonic potential $\frac{1}{2}k\mathbf{r}^2$ with spring constant k. Again it will be convenient to make a special module *force* for the computation of the forces. The arguments of this subroutine are the only potential parameter, namely the spring constant k, the number of atoms *natom*, the positions of the particles x, y, z , and the forces fx, fy, fz to be computed by the routine. We add a further output variable *epot* for the potential energy. The force routine could look like

```
subroutine force (k, natom, x, y, z, fx, fy, fz, epot)
epot := 0
do i := 1, natom
     fx(i) := -k * x(i)
     fy(i) := -k * y(i)
     fz(i) := -k * z(i)
     epot := epot + k * (x(i) * *2 + y(i) * *2 + z(i) * *2) /2
enddo
end
```

Code 4.16: Force routine for harmonic potential

epot is used here as a so called accumulator. The contribution to the potential of each particle is added to *epot* which on completion of the loop will contain the total potential energy. Since we will utilize $force$ repeatedly, the accumulator must be set to zero every time again on entering the force routine.

The parameters controlling the dynamics are the masses m_i of the particles, which are read from the array $ms(i)$, the time step δt represented by the constant dt and the number of steps *nstep* specifying the length of the run. What we choose for the value of the time step depends on a combination of factors, such as strength of the interactions, the mass and also the state of the system, i.e the temperature. In practice the time step is selected by a process of trial and error. We will return to this issue later on at length.

In the case of our harmonic potential an initial guess for the iteration step length is easier because there is a clearly defined time scale in our model, namely the period of oscillation $\tau = 2\pi\sqrt{m/k}$. This sets an upper limit to δt, and δt will be a fraction of τ. Exactly which fraction depends on the accuracy of the iteration scheme. For the Verlet algorithm $\delta t = \tau/20$ is a safe value. Still, this value may need some adjustment depending on the energy scale of the dynamics. Therefore, in the code examples below we will follow common practice, i.e will

leave the value of the time step a free parameter in the code which is assigned a definite value only at execution by reading it in from input. Other important system and control parameters such as the number of atoms and number of time steps will treated in the same way.

After all these preparations we are ready for the first version of the MD program.

```
program md_harmonic
read natom, l                                          \* read in system size *\
read nstep, dt                                  \* read time step and number of steps *\
read v0                                                \* read in initial velocity *\
call init_r (natom, l, x, y, z)                  \* initialize position on cubic grid *\
                                          \* uniform initial velocity in x direction *\
call init_v (natom, dt, v0, x, xm)
call init_v (natom, dt, 0, y, ym)
call init_v (natom, dt, 0, z, zm)
do mstep := 1, nstep
    call force (k, natom, x, y, z, fx, fy, fz, epot)                \* evaluate forces *\
    do i := 1, natom
                                  \* compute coordinates x(t + δt) according to Verlet *\
        xp := 2 * x(i) − xm(i) + dt * *2 * fx(i)/ms(i)
        yp := 2 * y(i) − ym(i) + dt * *2 * fy(i)/ms(i)
        zp := 2 * z(i) − zm(i) + dt * *2 * fz(i)/ms(i)
        xm(i) := x(i)                        \* x(t) → x(t − δt) update for next step *\
        ym(i) := y(i)
        zm(i) := z(i)
        x(i) := xp                           \* x(t + δt) → x(t) update for next step *\
        y(i) := yp
        z(i) := zp
    enddo
enddo
end
```

Code 4.17: Initializing and propagating positions according to Verlet

Note first of all that we now have two *do* loops, an outer loop, moving from one time iteration step to the next, and an inner loop, taking care of the advancement of the particle coordinates at iteration step *mstep*. The first three lines of the inner loop, cycling through the particle index, are a direct transposition of expression Eq. 4.23 for the $\mathbf{r}_i\,(t + \delta t)$ computed separately for the three components. The next 6 lines are updates preparing the $\mathbf{r}_i(t)$ and $\mathbf{r}_i\,(t - \delta t)$ for the next iteration step. Code 4.17 is the minimum Verlet code for coordinate iteration. Evaluation of the velocities $\mathbf{v}_i$, has been omitted since velocities are not needed in the Verlet algorithm. We may want to know $\mathbf{v}_i$, however, for the determination of properties depending on velocity. Kinetic energy (see Eq. 4.13) is one of these quantities. In fact, the value of the kinetic energy is crucial for the computation of two important quantities, namely the total energy (Eq. 4.14) and, as we shall see later, temperature. It is not difficult to obtain the kinetic energy from the variables in the inner loop of Code 4.17. Addition of one line is sufficient.

```
subroutine verlet (natom, dt, ms, fx, fy, fz, x, y, z, xm, ym, zm, ekin)
ekin = 0
```

```
do i := 1, natom
      xp := 2 * x(i) - xm(i) + dt * *2 * fx(i)/ms(i)
      yp := 2 * y(i) - ym(i) + dt * *2 * fy(i)/ms(i)
      zp := 2 * z(i) - zm(i) + dt * *2 * fx(i)/ms(i)
      ekin = ekin + ms(i) * ((xp - xm(i)) * *2 + (yp - ym(i)) * *2+
                                                 (zp - zm(i)) * *2)
      xm(i) := x(i)
      ym(i) := y(i)
      zm(i) := z(i)
      x(i) := xp
      y(i) := yp
      z(i) := zp
enddo
ekin = ekin/ (8 * dt * *2)
end
```

Code 4.18: Basic Verlet subroutine

The squares of the Verlet estimator Eq. 4.24 for velocities times the particle mass are accumulated in *ekin*, which, after adding in the last particle, is multiplied by the appropriate factors to convert it to the total kinetic energy Eq. 4.13. We also have packaged the code in a subroutine, for use as module in a full MD program. The arguments of subroutine Code 4.18 are: the constants needed in the Verlet loop, i.e. number of atoms *natom*, time step *dt* and masses *ms*, the components fx, fy, fz of the forces, which are supplied by a separate force subroutine *force*(see below), the current and previous coordinates, x, y, z respectively xm, ym, zm and kinetic energy *ekin*. On exit of the subroutine, the current and previous positions have been been advanced by one time step $t \rightarrow t + \delta t$, whereas *ekin* holds the value of kinetic energy at time t.

Our basic MD program Code 4.17 has now been reduced to specification of constants and a sequence of subroutine calls and some further operations, such as writing to output.

```
program md_harmonic
read natom, l, k
read nstep, dt
read v0
call init_r (natom, l, x, y, z)
call init_v (natom, dt, v0, x, xm)
call init_v (natom, dt, 0, y, ym)
call init_v (natom, dt, 0, z, zm)
do mstep := 1, nstep
      call force (k, natom, x, y, z, fx, fy, fz, epot)
      call verlet (natom, dt, ms, fx, fy, fz, x, y, z, xm, ym, zm, ekin)
      etot = ekin + epot
      print mstep, ekin, epot, etot
enddo
end
```

Code 4.19: Basic molecular dynamics in modular form

The kinetic energy *ekin* and potential energy *epot* fluctuate with the step number *mstep*. The sum of these two energies, the total energy *epot*, should be invariant since it is a constant of motion (see Eq. 4.14). The *print* command in Code 4.19 serves, therefore, an important purpose: it allows us to monitor the performance of the iteration so we can correct mistakes. One of the mistakes we could have made is taking a value for the time step which is too large, i.e exceeds the limits of reliability of the integration algorithm. This immediately shows in a drift, or even divergence of the total energy. In that case we have to stop the run, reduce the time step and start again. Practical rules for tolerance of total energy fluctuations are the basic tool for adjusting the time step, or for detecting bugs in a code.

Velocity Verlet

Finally we present the pseudo code giving the outline of a popular implementation of the velocity algorithm of Eqs. 4.25 and 4.29. Now the real particle velocities $\mathbf{v}_i(t)$ are used instead of the previous position $\mathbf{r}_i(t - \delta t)$. These velocities are stored in the arrays vx, vy and vz. To accommodate the peculiar midway force update of velocity Verlet (Eq. 4.26) we rearrange the position and velocity propagation steps Eq. 4.25 respectively 4.29 and define the following two procedures: A simple first order update for position only requiring the velocities as input and no forces

```
subroutine update_r (natom, dt, x, y, z, vx, vy, vz)
do i := 1, natom
      x(i) := x(i) + dt * vx(i)
      y(i) := y(i) + dt * vy(i)
      z(i) := z(i) + dt * vz(i)
enddo
end
```

Code 4.20: Position update for velocity Verlet

and a (partial) velocity update which takes account of the forces

```
subroutine update_v (natom, dt, ms, fx, fy, fz, vx, vy, vz)
do i := 1, natom
      vx(i) := vx(i) + dt * fx(i)/(2 * ms(i))
      vy(i) := vy(i) + dt * fy(i)/(2 * ms(i))
      vz(i) := vz(i) + dt * fz(i)/(2 * ms(i))
enddo
end
```

Code 4.21: Velocity update for velocity Verlet

Note that the propagation of the velocities is only over half the time step δt. The procedures Code 4.20, Code 4.21 and the force routine (Code 4.16 for the example of the harmonic oscillator) are the building blocks of the basic velocity Verlet MD loop :

```
call force (k, natom, x, y, z, fx, fy, fz, epot)          \* Initialize forces *\
do mstep := 1, nstep
                                          \* compute velocities at t + δt/2 *\
      call update_v (natom, dt, ms, fx, fy, fz, vx, vy, vz)
                                          \* compute positions at t + δt *\
```

```
    call update_r (natom, dt, x, y, z, vx, vy, vz)
                                                    \* compute forces at t + δt *\
    call force (k, natom, x, y, z, fx, fy, fz, epot)
                                                    \* compute velocities at t + δt *\
    call update_v (natom, dt, ms, fx, fy, fz, vx, vy, vz)
enddo
end
```

Code 4.22: Basic velocity Verlet loop

We have omitted from Code 4.22 all initialization calls except for the forces which is a feature special to velocity Verlet.

Exercise 3: *Show that Code 4.22 is indeed equivalent to Eqs. 4.25 and 4.29. What could be the advantage of using an intermediate update at half time step for velocity?*

Project A: *Write a Verlet or velocity Verlet code (whatever you prefer) for the 3 dimensional harmonic oscillator.*

$$v(\mathbf{r}) = \frac{k}{2}\mathbf{r}^2 \tag{4.58}$$

Use reduced units defined by setting $k = 1$ and the unit of particle mass to 1.

i Plot for a single particle of unit mass the potential energy, kinetic energy and total energy as a function of time for various values of time step and initial conditions. Explain what you see.

ii Plot for 10 particles of different mass the potential energy, kinetic energy and total energy as a function of time. Give the particles different initial positions and velocities. Explain.

4.5.3 ∗Temperature control

Velocity scaling

Adding the simple temperature scaling scheme of Eq. 4.50 is simple. We will again introduce a separate subroutine for this purpose. The arguments of this subroutine, called *tmp_scale*, include the set thermodynamic temperature *tmpset* (T in Eq. 4.50) which is a parameter read in, together with other constants, at the start of the program.

```
subroutine tmp_scale (natom, dt, ms, tmpset, x, y, z, xm, ym, zm)
ekin := 0
do i := 1, natom                                    \* determine kinetic energy *\
    ekin := ekin + ms(i) * ((x(i) − xm(i)) * *2 + (y(i) − ym(i)) * *2+
                            (z(i) − zm(i)) * *2)
enddo
                                                    \* convert to instantaneous temperature *\
tmpkin := ekin/ (3 * natom * dt * *2)
fact := sqrt (tmpset/tmpkin)                        \* scaling factor for velocities *\
                                                    \* scale velocities by adjusting r_i (t − δt) *\
do i := 1, natom
    xm(i) := x(i) − fact * (x(i) − xm(i))
    ym(i) := y(i) − fact * (y(i) − ym(i))
    zm(i) := z(i) − fact * (z(i) − zm(i))
```

```
enddo
end
```

Code 4.23: Temperature scaling procedure for Verlet

Velocity is not explicitly used as iteration variable in the Verlet approach (Eq. 4.23). It is implicitly represented by the increment in position $\mathbf{r}(t) - \mathbf{r}(t - \delta t)$ made by the previous time step. Therefore, in order to apply velocity scaling, either $\mathbf{r}(t)$ or $\mathbf{r}(t - \delta t)$ must be modified. In Code 4.23 the past $\mathbf{r}(t-\delta t)$ is redefined. In view of the force calculation, this is more convenient than changing the present. One more comment on Code 4.23 concerns the question of the choice of units. In order to avoid repeated multiplication by conversion factors we have opted for using atomic units as internal program units. This implies that both the kinetic temperature and set temperature in Code 4.23 are in Hartree, the atomic unit of energy. Therefore, the parameter *tmpset* must have been converted from practical temperature units, Kelvin, to atomic units at the beginning of the program. We will defer further discussion of this issue till section 4.5.5.

The next piece of code shows where to insert the call to *tmp_scale* in the MD loop of Code 4.19

```
do mstep := 1, nstep
    call force (k, natom, x, y, z, fx, fy, fz, epot)
                                    \* scale temperature once every nscale steps *\
    if (mod(mstep, nscale) = 0) then
        call tmp_scale (natom, dt, ms, tmpset, x, y, z, xm, ym, zm)
    endif
    call verlet (natom, dt, ms, fx, fy, fz, x, y, z, xm, ym, zm, ekin)
    etot = ekin + epot
    tmpkin := ekin/ (3 * natom)
    print mstep, tmpkin, epot, etot
enddo
```

Code 4.24: Verlet MD loop with periodic temperature scaling

Scaling temperature every time step is not necessary and, in fact, not desirable either. Therefore, the *tmp_scale* procedure in Code 4.24 is called only periodically, namely once every *nscale* steps. This is achieved by the $mod(mstep, nscale) = 0$ condition in the *if* statement. The modular function *mod* is defined in Code 4.11.

Velocity Initialization

In this section we will outline a method for the initialization of velocity using a random number generator routine. Entering the first iteration step of the MD loop in Code 4.24, the velocities still have the uniform initial drift in the x direction we have set up in subroutine Code 4.19. For the study of thermal systems, it would be more convenient, and save computer time, if the velocities already would have a thermal distribution of initial values appropriate for thermal equilibrium at *tmpset*. The canonical probability distribution $P(v_{x,i})$ for the x components of velocity of particle i is a simple Gaussian given in Eq. 4.52. Initialization of velocities amounts, therefore, to sampling the Gaussian probability distribution of Eq. 4.52.

There are several methods for converting random numbers of uniform probability into numbers distributed according to a Gaussian. We will sketch here a simple approach, which may not be the most efficient in terms of computer time, but is good enough for velocity initialization.

```
subroutine init_v (natom, dt, iseed, x, xm)
do i := 1, natom
      vrndm := −6.0
      call my_ran (12, rndm, iseed)
      do m := 1, 12
           vrndm := vrndm + rndm(i)
      enddo
      xm(i) := x(i) − dt ∗ vrndm
enddo
end
```

Code 4.25: Sampling a velocity component from Gaussian with zero mean and unit width

Exercise 4: *Code 4.25 is based on the Gaussian statistics of random walks (only of length 12 in this case). Proof that the variance is indeed unity. For a better algorithm for the generation of Gaussian random numbers see for example appendix H of FS.*

This routine *init_v* will from now on replace the temporary version of Code 4.15 for velocity initialization. The complete set of initialization calls, placing the particles on a cubic grid and giving them velocities in random directions sampled at temperature $T = tempset$ is now

```
call init_r (natom, l, x, y, z)
call init_v (natom, dt, iseed, x, xm)
call init_v (natom, dt, iseed, y, ym)
call init_v (natom, dt, iseed, z, zm)
call tmp_scale (natom, dt, ms, tmpset, x, y, z, xm, ym, zm)
```

Code 4.26: setting up particles in a cubic box with thermalized velocities

The call to *tmp_scale* of Code 4.23 transforms the velocities sampled from a Gaussian with unit width to the values consistent with the temperature and particle mass.

Project B: *Write the code for the initialization calls in Code 4.26 and test this out for the ten particles in the 3D harmonic well of Project A. Choose a medium temperature, say 1 or 2 in reduced units and plot the kinetic temperature, the potential energy and total energy as a function of time for a run with a couple of temperature scaling operations. What is the average variation in total energy?*

4.5.4 ∗Force computation

Basic force loop

Having introduced the basic procedures of an MD code, we now want to replace the harmonic potential in the force routine of Code 4.16 by an interacting potential, for which we take the 12-6 Lennard Jones potential of Eq. 4.53. The spherical cutoff Eq. 4.54 can be very compact. The *force* routine of Code 4.16 is modified to

```
subroutine force(fmodel, l, rc, natom, x, y, z, fx, fy, fz, epot)
feps := 4 ∗ fmodel(1)                    \∗ model_parameter_1 = ε → feps = 4ε ∗\
sigm2 := fmodel(2) ∗ ∗2                  \∗ model_parameter_2 = σ → sigm2 = σ ∗ ∗2 ∗\
rc2 = rc ∗ ∗2                                          \∗ square of cutoff radius r_c ∗\
```

```
do i := 1, natom                          \* setting force accumulators to zero *\
    fx(i) := 0
    fy(i) := 0
    fz(i) := 0
enddo
epot := 0                                 \* setting potential energy accumulator to zero *\
do i := 1, natom - 1
    do j := i + 1, natom                  \* interactions of pairs i, j with j > i *\
        dx := x(j) - x(i)
        dy := y(j) - y(i)
        dz := z(j) - z(i)
        dr2 := dx * *2 + dy * *2 + dz * *2
                                  \* skip particles j outside cutoff sphere around particle i *\
        if (dr2 ≤ rc2) then
            dr2nv := 1/dr2
            rm6 := (sigm2 * dr2nv) * *3
            rm12 := rm6 * *2
            epot := epot + feps * (rm12 - rm6)
            frad := feps * dr2nv * (12 * rm12 - 6 * rm6)
                                  \* 1/r× radial force: frad = -1/r d/dr v(r) *\
            fx(j) := fx(j) + frad * dx    \* add to total force on particle j *\
            fy(j) := fy(j) + frad * dy
            fz(j) := fz(j) + frad * dz
            fx(i) := fx(i) - frad * dx    \* add to total force on particle i *\
            fy(i) := fy(i) - frad * dy
            fz(i) := fz(i) - frad * dz
        endif
    enddo
enddo
end
```

Code 4.27: Force routine for 12-6 potential with spherical cutoff

To give the force routine a generic form that can be used for other force fields as well, the parameters specifying the force field are passed to the subroutine by means of an array *fmodel* in the argument. In this case *fmodel* has two elements, namely ϵ and *sigma* converted to atomic units, the internal units of our code (see comment section 4.2.3). The second argument of *force* is the length l of an edge of the cubic MD cell. We will need this parameter later for the application of periodic boundary conditions. The third argument is radius r_c of Eq. 4.54 of the cutoff sphere. The force calculation is the part of the MD program taking most of the CPU time. Optimization of the code for the force loop is, therefore, most critical. Two features of Code 4.27, which may seem somewhat odd at first, have been motivated by reduction of the number of numerical operations. First, the lower boundary of the second loop over index j for a given value of i ensures that i, j pairs that have already been considered previously for lower values of i are not included again. This means that we must update the forces for both i and j using Newton's third law $f_{ji} = -f_{ij}$. Second, the computation of the square root $r = \sqrt{r^2}$ is avoided. The square root is a relatively expensive operation. This was particularly

noticeable for the generation of computers in the 60's and early 70's on which these MD codes were developed.

Periodic boundaries
We will illustrate the code for the minimum image approximation (Fig. 4.3) for a cubic box, i.e $\mathbf{a}, \mathbf{b}, \mathbf{c}$ have all the same length L and are directed along the x respectively y and z axis of the Cartesian frame. Again, we will construct a special module for the purpose of determining the neighboring image vector dx, dy, dz of a pair of particles with coordinates $x1, y1, z1$ and $x2, y2, z2$ in a MD box with edge $L = l$

```
subroutine pbc(l, x1, y1, z1, x2, y2, z2, dx, dy, dz)
dx = x2 − x1
dy = y2 − y1
dz = z2 − z1
if (l > 0) then
      dx = dx − l ∗ nint (dx/l)
      dy = dy − l ∗ nint (dy/l)
      dz = dz − l ∗ nint (dz/l)
endif
end
```

Code 4.28: Minimum image vector for simple cubic periodic boundary conditions

The action of the intrinsic function *nint* was defined in Code 4.12. We have verified already in exercise 1 that the operation $y = x - l * nint\,(x/l)$ indeed reduces x to a number y with magnitude $|y| \leq l/2$ and the correct sign. The *if* statement in Code 4.28 is a safeguard against dividing by zero and at the same time enables us to use the *pbc* routine even when we don't want to apply periodic boundary conditions. In that case we simply set l to a zero or negative value.

Implementation of periodic boundary conditions in the minimum approximations in the *force* routine of Code 4.27 is straight forward. All we have to do is to replace the computation of the coordinate differences $x(j) - x(i)$, etc by a call to the *pbc* routine of Code 4.28.

```
do i := 1, natom − 1
      do j := i + 1, natom
            call pbc(l, x(i), y(i), z(i), x(j), y(j), z(j), dx, dy, dz)
            dr2 := dx ∗ ∗2 + dy ∗ ∗2 + dz ∗ ∗2
            if (dr2 ≤ rc2) then
                  ···                  \∗ calculations of forces and energies, see Code 4.27 ∗\
            endif
      enddo
enddo
```

Code 4.29: Force loop with periodic boundary conditions in minimum image approximation

4.5.5 ∗The MD code for liquid argon

The modules of sections 4.5.2, 4.2.3 and 4.3.2 are the building blocks for a minimal but complete MD code for the simulation of liquid Argon. In this section we will outline how this code can

be set up and also mention some further practicalities.

Main program plus input
The core MD program for a monoatomic liquid is fairly straightforward. The program consists of an initialization part, the actual MD loop, followed by some code for conversion and analysis. Below is a complete code doing all basic tasks.

```
program md_argon
read dt, nstep                                   \* time step, number of steps *\
read restart, averages, nsave                    \* restart and save instructions *\
read tmpset, iseed, nscale                       \* temperature control *\
read natom, lbox                                 \* system size parameters *\
read epsilon, sigma, rc                          \* interaction parameters and spherical cutoff *\
                                                 \* conversion to internal units and defaults *\
call const_md (natom, clen, ctim, cenr, ctmp,
               lbox, rc, dt, tempset, epsilon, sigma, fmodel, ms)
if (restart) then
                                                 \* read information for continuation of previous run *\
    call read_md (natom, lbox, x, y, z, xm, ym, zm, lstep, sekin, sepot)
else                                             \* or start new run *\
    call init_r (natom, lbox, x, y, z)
    call init_v (natom, dt, iseed, x, xm)
    call init_v (natom, dt, iseed, y, ym)
    call init_v (natom, dt, iseed, z, zm)
    call tmp_scale (natom, dt, ms, tmpset, x, y, z, xm, ym, zm)
    averages := .false.
endif
if (averages) then                               \* continue averaging *\
    nstep := lstep + nstep
else                                             \* or reset accumulators *\
    lstep := 0
    sekin := 0
    sepot := 0
endif
do mstep := lstep + 1, nstep
    call force (fmodel, lbox, rc, natom, x, y, z, fx, fy, fz, epot)
    if (mod(mstep, nscale) = 0) then
        call tmp_scale (natom, dt, ms, tmpset, x, y, z, xm, ym, zm)
    endif
    call verlet (natom, dt, ms, fx, fy, fz, x, y, z, xm, ym, zm, ekin)
    sekin := sekin + ekin                        \* update energy accumulators *\
    sepot := sepot + epot
                                                 \* periodic backup of configuration and accumulators for averages *\
    if (mod(mstep, nsave) = 0) then
        call write_md (natom, lbox, x, y, z, xm, ym, zm, mstep, sekin, sepot)
    endif
                                                 \* conversion to practical units and output for monitoring performance *\
    ekin := cenr * ekin
    epot := cenr * epot
```

```
        etot := ekin + epot
        tmpkin := ctmp * ekin/(3 * natom)
        print mstep, tmpkin, epot, etot
    enddo
    sepot := cenr * sepot/(nstep)          \* compute and print averages over full run *\
    stmpkin := ctmp * cenr * sekin/(nstep * 3 * natom)
    print nstep, stmpkin, sepot
    end
```

Code 4.30: Outline of a MD program for liquid argon

There are several new features in Code 4.30 dealing with more practical matters in a MD simulation, which we will explain in the next two sections. For completeness we also give a sample input file for liquid Argon.

```
\* DATA FILE FOR LIQUID ARGON *\
5, 1000                                    \* do 1000 time steps of length 5 fs *\
.true., .false.,200  \* restart previous run resetting averages, save every 200 steps *\
150.0,0,100    \* scale temperature every 100 steps aiming at an average of 150K *\
                                  \* default seed for temperature initialization *\
108, 18.0                        \* 108 atoms in a cubic box with side L = 18 Å*\
1.0, 3.4, 9.0          \* ε = 1 kJmol^-1, σ = 3.4Å, with spherical cutoff of L/2 *\
```

Code 4.31: Sample input file for a MD simulation of liquid argon

The question of units

Most MD and MC codes work with internal units which are different from the units read in from input. The subroutine *const_md* is a multipurpose routine called at the beginning of execution to convert input parameters to internal program units and assign default values to those parameters that have not been read in (such as the mass). It also makes the conversion factors it applied available for use later in the program. For LJ systems such as argon state variables and other quantities are usually converted to scaled units defined by making σ the unit of length, ϵ the unit of energy and the mass of an atom (40 in the case of argon) the unit of mass. Quantities in reduced units are indicated by stars. The frequently occurring scaled variables are listed below.

$$\begin{array}{lcll} \rho^* & = & \rho\sigma^3 & \text{density} \\ T^* & = & k_B T/\epsilon & \text{temperature} \\ P^* & = & P\sigma^3/\epsilon & \text{pressure} \\ t^* & = & \left(\epsilon/m\sigma^2\right)^{1/2} t & \text{time} \end{array} \tag{4.59}$$

These scaled units could also be used as internal program units. For more general force fields involving also electrostatic interactions, however, it is more convenient to use atomic units. It will be clear, though, that preferences for one type of internal units over another are somewhat subjective. Below we give an example of the code for the procedure *const_md* meant for the use of atomic units as internal units.

```
subroutine const_md (natom, clen, ctim, cenr, ctmp,
                     lbox, rc, dt, tempset, epsilon, sigma, fmodel, ms)
clen := 1.889763                                   \* Angstrom to a.u. *\
```

```
ctim := 41.33983                                  \* femtosecond to a.u. *\
cenr := 2625.4                                    \* a.u. to kJmol^-1 *\
ctmp := 120.28                                    \* kJmol^-1 to Kelvin *\
cmas := 1822.92                                   \* a.u. of nuclear mass *\
lbox := clen * lbox
rc := clen * rc
dt := ctim * dt
tempset := tempset/ (ctmp * cenr)
fmodel(1) := epsilon/cenr
fmodel(2) := clen * sigma
argmass := 39.95 * cmas
do i := 1, natom
    ms(i) := argmas
enddo
end
```

Code 4.32: Conversion to atomic program units and default initialization

Restarting from a previous run

Next we come to a very useful device in the practice of MD simulation, namely the restart option. Since the motion in MD reflects the physical evolution of the system, equilibration is an important and expensive part of simulation. In view of this, we certainly don't want to start every run again from scratch, but be able to continue from the point where we terminated a previous run. This is the motivation behind the periodic backup of the instantaneous configuration of the system by the subroutine *write_md* in the MD loop of Code 4.30. In the Verlet algorithm the dynamical state of a system is given by the current and previous positions. The procedure *write_md* writes these arrays to an external file which we will call RESTART. These data are read in at the beginning of the next run by the subroutine *read_md* provided we have requested the program to so by setting the control parameter *restart* to ".true.". The code for *write_md* in its simplest form could be as follows

```
subroutine write_md (natom, lbox, x, y, z, xm, ym, zm, mstep, sekin, sepot)
write (RESTART) lbox,natom
write (RESTART) x,y,z,xm,ym,zm
write (RESTART) mstep,sekin,sepot
end
```

Code 4.33: writing to a restart file

The *write* command in Code 4.33 differs from the *print* statement used so far, because now, instead of writing to standard output (e.g. the terminal screen), we want to write to a specific file which is identified to the *write* command by giving its name as an argument (in real computer languages a *write* statement is considerably more complicated requiring several further parameters). Similar rules apply for the corresponding *read* command in the subroutine *read_md* in the code below.

```
subroutine read_md (natom, lbox, x, y, z, xm, ym, zm, lstep, sekin, sepot)
read (RESTART) lboxx,natomx
if (natomx ≠ natom) then
    print "ERROR: wrong number of atoms"
```

```
    stop
endif
if (lboxx ≠ lbox) then
    print "WARNING: box length not the same"
endif
read (RESTART) x,y,z,xm,ym,zm
read (RESTART) lstep,sekin,sepot
end
```

Code 4.34: reading from a restart file with some precautions

x, y, z and xm, ym, zm represent all the information we need to restart the dynamics. Therefore, possible data previously stored on RESTART can be overwritten. With no record of the past other then the last step, we are, of course, not able to continue with the accumulation of averages. Thus, the accumulators together with the total number of accumulated steps are attached at the end of the RESTART file by Code 4.33. They are also read in by *read_md* in Code 4.34. If we wish to continue with these accumulators we give the control parameter *averages* the value ".true.". If, on the other hand, *averages* = .false. the accumulators are reset to zero. The logic for these operations in Code 4.30, including the manipulations of the *lstep* counter, should be self-explanatory.

The system data *natom* and *lbox* in the first record of the RESTART file are there to prevent mistakes. Restarting from the wrong file belonging to a system with a different number of particles is usually catastrophic. Also a change of box size, while not fatal, can lead to major instabilities. The subroutine *read_md* contains some logic to check for inconsistencies of data read in from input and the data on RESTART. If Code 4.34 detects that the number of particles is not the same, it writes an error message to standard output and aborts execution. This is what the *stop* command does. If the box size is different, the routine only issues a warning. A change in box size is what we may have intended for example when we want to adjust the density. However, one should always take the utmost most care with such operations, because the system in general will respond rather violently to changes in boundary conditions.

Project C: *Implement the MD code for argon of Code 4.30, starting from scratch or from the code you have made for projects A and B. Set up a system of 108 argon atoms in a cubic box in the thermodynamic state specified in reduced units(Eq. 4.59) by a density of* $\rho^* = 0.844$ *and a temperature of* $T^* = 1.5$ *(corresponding to 180 K). This is not far from the triple point. Use a cutoff for the interactions of half the boxlength. As a rule of thumb total energy is not allowed to drift in the fourth digit over thousand time steps if no temperature scaling is applied. How large can you make the time step before energy conservation starts to suffer?*

Chapter 5

Probing static properties of liquids

5.1 Liquids and simulation

The molecular dynamics (MD) method was developed in the late 1950's to understand the physics of mono-atomic (noble gas) liquids. What is so special about liquids? Unlike solids the atoms in liquids have no fixed equilibrium position (they diffuse). As a result there is *no long range order* in the form of a lattice. That is true for gases as well. However, unlike gases the density in liquids is high, only a little less than in the corresponding solid (and sometimes higher, water!). The atoms are therefore strongly interacting, establishing a local environment very similar to solids, i.e liquids exhibit solid-like *short range order*. Clearly finite temperature is crucial for stabilization of this dynamical disorder in liquids keeping them from freezing. So, the first contribution of computer simulation was to provide accurate "experimental" data for model systems for which the interactions were precisely known (e.g. the 12-6 Lennard-Jones potential Eq. 4.53). This chapter is an introduction of some of the statistical methods used to abstract data from a simulation that give insight in structure and, at the same time can be compared to real experimental data. Characterization of dynamics is discussed in Chapter 3.

5.2 Radial distribution function

5.2.1 Radial distribution function

Liquids are homogeneous systems with an on average uniform particle density $\rho(\mathbf{r}) = \rho = N/V$, where N is the number of atoms in a container with volume V, or, when dealing with a computer model, N is the number of atoms in a periodic cell with volume V. How to probe the atomic scale local structure in liquids? A statistical quantity defined for just this purpose is the pair correlation or radial distribution function. As it turns out, pair correlations are also accessible by experiments, the most direct way is the measurement of structure factors by neutron or X-ray scattering experiments.

Radial distribution functions are essentially histograms of two-particle distances. This "operational definition" is perhaps the most convenient approach for a first introduction. So we will go through the steps for constructing such a distance histogram.

i Pick a reference particle i with position $\mathbf{r}_i$

ii Draw a spherical shell of radius r and thickness Δr with center at $\mathbf{r}_i$

iii A particle j in this shell has a distance $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$ w.r.t. i

$$r - \Delta r \leq r_{ij} < r$$

Determine the number of particles in the shell, call this number $n_i(r, \Delta r)$

iv Divide by the volume of the shell and average over reference particles

$$\Rightarrow \frac{1}{N} \sum_i^N \frac{n_i(r, \Delta r)}{4\pi r^2 \Delta r}$$

v Normalizing by the particle density $\rho = N/V$ we obtain

$$g(r) = \frac{V}{4\pi r^2 \Delta r N^2} \sum_i^N n_i(r, \Delta r) \tag{5.1}$$

For "sufficiently" small width Δr, this is our estimate of the radial distribution function (RDF) which is therefore a dimensionless quantity. This procedure is illustrated in Fig: 5.1 for the example of a 2 D Lennard-Jones fluid.

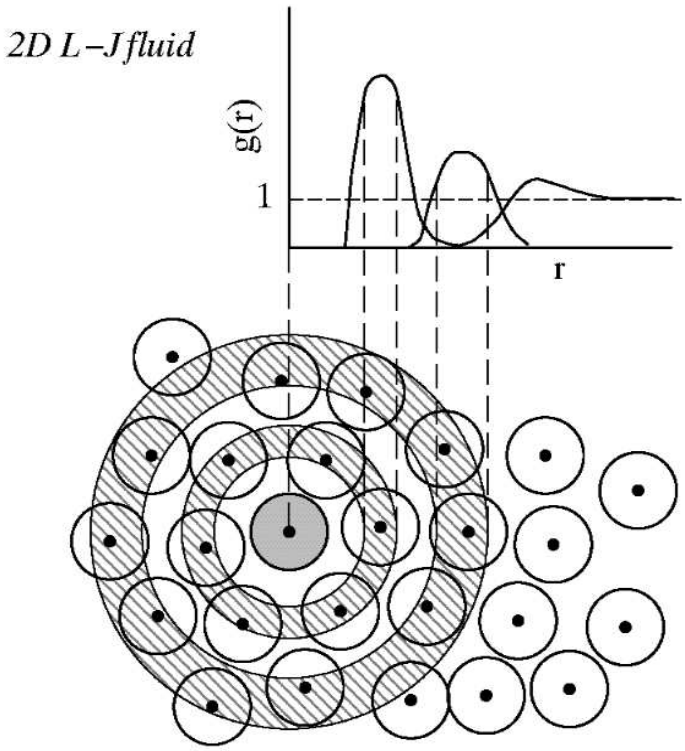


Figure 5.1: Coordination and radial distribution function in a two dimensional fluid of disks. The first shell (nearest neighbours) gives rise to the first peak in the RDF. The shallow second peak is the result of the more loosely coordinated 2-nd coordination shell (2nd nearest neighbours). For larger distances the correlation is lost (homogeneous liquid)

For distances $r < \sigma$, where σ is the repulsive (hard) core diameter, the radial distribution vanishes because particles are excluded from this region.

$$g(r < \sigma) = 0 \tag{5.2}$$

The maximum at a distance a little over σ reflects the well defined coordination shell of nearest neighbours around a particle in a liquid. This peak in the $g(r)$ is characteristic for the high densities prevalent in liquids and is absent in the vapour phase. In most liquids there is also a broad second nearest neighbour peak. As a result of the disorder in a liquid, this structure is considerably less pronounced compared to (high temperature) solids of similar densities (see further section 5.2.3). For distances larger than second neighbours (a multiple of σ) fluctuations take over and the distribution of atoms as seen by the reference particle approach the homogeneous limit of the liquid. That means that for large r the number of particles $\Delta n(r, \Delta r)$ in shell $r - \Delta r \leq r' < r$

$$\frac{\Delta n(r, \Delta r)}{4\pi r^2 \Delta r} \approx \frac{N}{V}$$

which gives, when substituted in the definition of the RDF

$$g(r) = \frac{V}{N^2} \sum_i^N \frac{n_i\left(r, \Delta r\right)}{4\pi r^2 \Delta r} \approx \frac{V}{N^2} \sum_i^N \frac{N}{V} = 1$$

Thus, the radial distribution function approaches unity for distance larger than some characteristic correlation length ξ

$$g\left(r > \xi\right) = 1 \tag{5.3}$$

In MD or MC codes radial distribution functions are estimated by making a histogram of particle distances. The accumulation of such a histogram could be carried out as part of the force loop where all pair distances have been made available for the computation of the forces. An illustration of such a combined force and RDF loop is given in appendix 5.4.1, where also the somewhat tricky normalization of the histogram converting it to a true RDF is discussed.

5.2.2 Coordination numbers

As suggested by Fig. 5.1 the integral of the first peak of the RDF is related to the average number of particles n_c in the first coordination shell. This is generally true. In order to measure n_c we must specify how close an atom must approach the central particle in order to be counted as a first neighbour. The position r_{min} of the minimum of the first and second maximum is used as a common (but not unique) criterion to define the first coordination shell. n_c is then found from the integral (in three dimensional space)

$$n_c = 4\pi\rho \int_0^{r_c} dr\, r^2 g(r) \tag{5.4}$$

with the $r_c = r_{min}$. For general values of the radius r_c the integral n_c is the called the coordination number giving the average number of atoms within a distance r_c from the reference particle. By way of proof of Eq. 5.4 we again go through to steps of the computation coordination numbers from the histogram of pair distances. Summing the number of particles $n_i\left(r_m\right)$ in shells $r_{m-1} < r \leq r_m + \Delta r$ surrounding particle i (see Eq.5.1) upto shell M starting at m=1 (i.e the origin) gives

$$N_i\left(r_M\right) = \sum_{m=1}^{M} n_i\left(r_m\right)$$

$N_i(r_M)$ is therefore the number of particles within distance $r_M = M\Delta r$ from particle i. The average over reference particles i

$$n_c(r_M) = \frac{1}{N}\sum_{i=1}^{N} N_i(r_M)$$

is the coordination number for radius $r_c = M\Delta r$. Rearranging the summations over particle index i and bin index m

$$n_c(r_M) = \frac{1}{N}\sum_{i=1}^{N} N_i(r_M) = \frac{1}{N}\sum_{i=1}^{N}\sum_{m=1}^{M} n_i(r_m) = \sum_{m=1}^{M}\left[\frac{1}{N}\sum_{i=1}^{N} n_i(r_m)\right]$$

we can substitute the definition for the RDF (Eq. 5.1) to find

$$n_c(r_M) = \sum_{m=1}^{M}\frac{N}{V}\,4\pi\Delta r\, r_m^2 g(r_m)$$

Replacing summation over shells by a spherical integral gives Eq. 5.4(for small Δr).

5.2.3 Examples of radial distribution functions

As an illustration of how RDF's are used to characterize the local environment in a liquid it is instructive to compare the liquid to the solid under similar conditions. Fig. 5.2 shows the RDF of argon for the liquid and solid near the triple point. For the solid the RDF is highly structured showing sharp distinct peaks. The RDF in the liquid is a "washed out" version of this. In order to understand these features, let us first consider the RDF of an ideal low

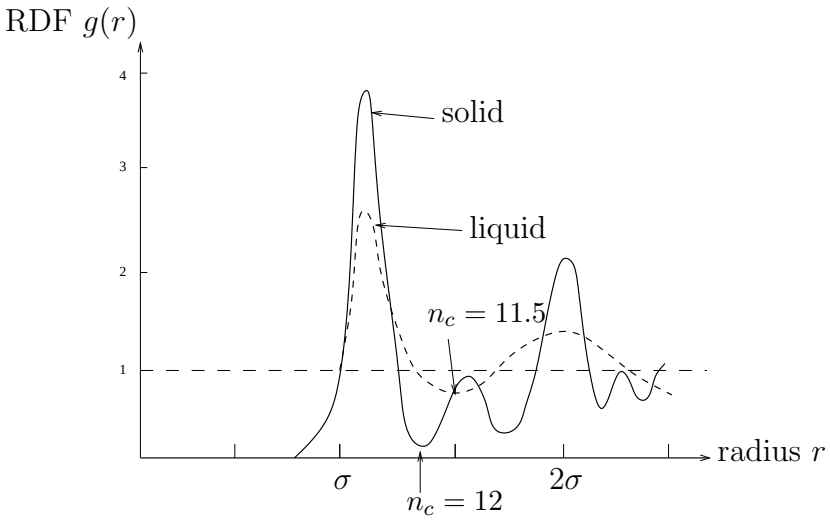


Figure 5.2: RDF for liquid (dashed curve) and solid argon (solid curve) at the triple point compared.

temperature fcc crystal schematically indicated in Fig. 5.3. Because the temperature is low the peaks are only slightly broadened by vibrational motion (phonons). So, the first nearest neighbour peak, at $r = \sigma$, stands out and is clearly separated from the second at $r = \sqrt{2}\sigma$. The

ratio of $\sqrt{2}$ is dictated by the geometry of closed packed lattices (fcc and hcp) in 3 dimensions (the fcc lattice is reproduced in Fig. 5.4) and so is the number of nearest neighbours (12) and next nearest neighbours (6). The coordination number at $r = \sqrt{2}\sigma$ is therefore $n_c = 12$ going up to $n_c = 18$ at $r = \sigma$ when the 6 next nearest neighbours have been added.

In spite of the much higher high temperature (the solid is close to melting) we can still recognize the outlines of a close packed fcc arrangement in the first three peaks in Fig. 5.2. The first (nearest neighbour) peak is preserved in the liquid in an even more smeared out form with a marginally smaller coordination number (11.5) when measured at the first minimum (Fig. 5.2), indicating that for correlations in the first coordination shell in liquids are indeed similar to solids. The second and third maximum, however, have become fused in a broad hump as a result of the accumulating disorder at larger distances.

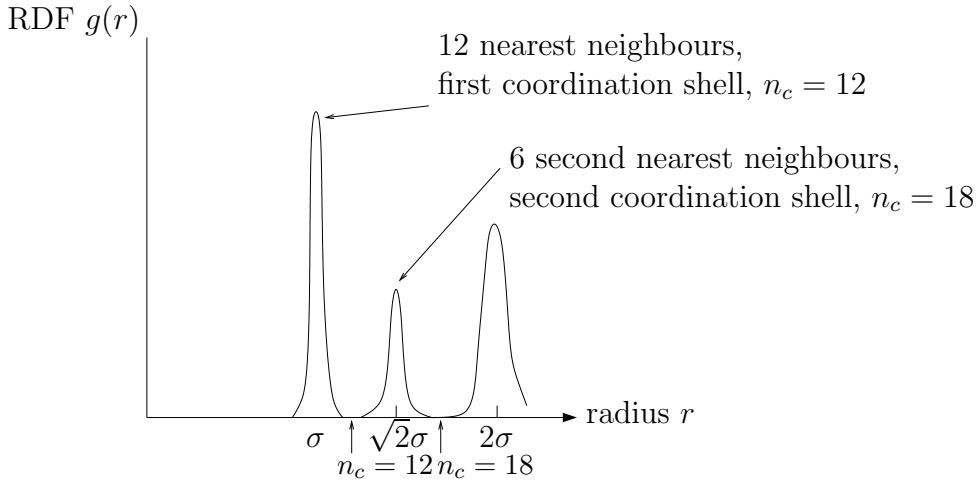


Figure 5.3: RDF of a face centered cubic (fcc) mono-atomic solid with coordination numbers at selected points (arrows). σ is the hard core diameter for pair interactions.

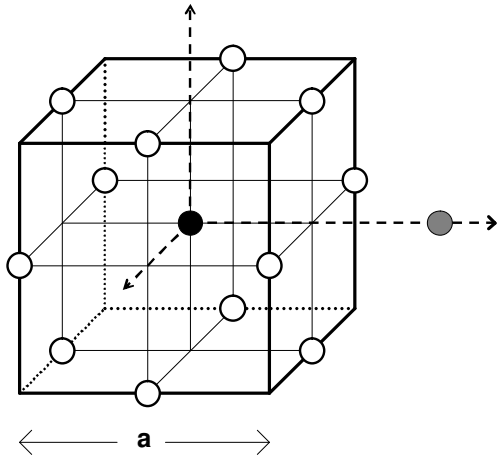


Figure 5.4: fcc lattice. Distance to the 12 nearest neighbours (white balls) is $a/\sqrt{2}$. The 6 next nearest neighbours are at a distance a, only one of them is indicated (grey ball).

5.2.4 Radial distribution function in statistical mechanics

The way we introduced the radial distribution function in Eq. 5.1 was "operational" in the sense that it is based on how this quantity is determined in a simulation. We will now give a proper statistical mechanical definition of the RDF making use of Dirac delta functions:

$$g\left(\mathbf{r}\right)=\frac{V}{N^{2}}\left\langle \sum_{i,\,j\neq i}^{N}\delta\left(\mathbf{r}_{ij}-\mathbf{r}\right)\right\rangle \tag{5.5}$$

where the angular brackets denote an integral over the configurational probability distribution function of Eq. 4.44, containing all configurational information there is to know. So, in explicit form the function $g\left(\mathbf{r}\right)$ is written as

$$g\left(\mathbf{r}\right)=\frac{V}{N^{2}}\sum_{i,\,j\neq i}^{N}\int d\mathbf{r}_{1}d\mathbf{r}_{2}\ldots d\mathbf{r}_{N}P_{N}\left(\mathbf{r}^{N}\right)\delta\left(\mathbf{r}_{ij}-\mathbf{r}\right) \tag{5.6}$$

Note the different role of the vectors in the argument of the delta function. $\mathbf{r}_{ij}=\mathbf{r}_{i}-\mathbf{r}_{j}$ are particle coordinates and also appear as integration variables in the statistical average (Eq. 5.6). The vector $\mathbf{r}$, on the other hand, is parameter with a value of our choice and thus appears as a true argument on the left hand side. $g\left(\mathbf{r}\right)$ has a direct probabilistic interpretation. It is proportional to the probability for observing any two particles separated by a vector $\mathbf{r}$ (proportional but identical because of the difference in normalization factor). Liquids are isotropic systems with no preference for a direction in space. We therefore expect the function $g\left(\mathbf{r}\right)$ of Eq. 5.5 to depend only on the length $r=\left|\mathbf{r}\right|$. Using this property we integrate over a shell $V(r,\Delta r)$ between r and $r+\Delta r$ and, assuming that the shell is sufficiently thin, we write

$$\int_{V(r,\Delta r)}d\mathbf{r}\,g\left(\mathbf{r}\right)\approx4\pi r^{2}\Delta rg(r) \tag{5.7}$$

where the function $g(r)$, to be identified with the RDF, is now a function of radial distance only. Substituting in Eq. 5.6 we have

$$\begin{aligned}g(r)\approx&\frac{V}{4\pi r^{2}\Delta rN^{2}}\int_{V(r,\Delta r)}d\mathbf{r}\sum_{i,\,j\neq i}^{N}\int d\mathbf{r}_{1}d\mathbf{r}_{2}\ldots d\mathbf{r}_{N}P_{N}\left(\mathbf{r}^{N}\right)\delta\left(\mathbf{r}_{ij}-\mathbf{r}\right)\\=&\frac{V}{4\pi r^{2}\Delta rN^{2}}\int d\mathbf{r}_{1}d\mathbf{r}_{2}\ldots d\mathbf{r}_{N}P_{N}\left(\mathbf{r}^{N}\right)\sum_{i,\,j\neq i}^{N}\int_{V(r,\Delta r)}d\mathbf{r}\,\delta\left(\mathbf{r}_{ij}-\mathbf{r}\right)\end{aligned} \tag{5.8}$$

where in the second equation we have interchanged the order of the integration, integrating the delta functions over the parameter $\mathbf{r}$ first. This gives unity when the vector $\mathbf{r}_{ij}=\mathbf{r}_{i}-\mathbf{r}_{j}$ lies within volume $V(r.\Delta r)$ and zero otherwise. The delta function therefore counts the number of particle pairs inside the shell. Taking particle i as reference we recover the quantity $n_{i}\left(r,\Delta r\right)$ introduced in the scheme leading to Eq. 5.1

$$n_{i}\left(r,\Delta r\right)=\sum_{j\neq i}\int_{V(r,\Delta r)}d\mathbf{r}\,\delta\left(\mathbf{r}_{ij}-\mathbf{r}\right) \tag{5.9}$$

Substituting in Eq. 5.7 we find

$$\begin{aligned} g(r) \approx & \frac{V}{4\pi r^2 \Delta r N^2} \int d\mathbf{r}_1 d\mathbf{r}_2 \ldots d\mathbf{r}_N P_N\left(\mathbf{r}^N\right) \sum_i n_i(r, \Delta r) \\ = & \frac{V}{4\pi r^2 \Delta r N^2} \left\langle \sum_i n_i(r, \Delta r) \right\rangle \end{aligned} \tag{5.10}$$

which is the expectation value of the instantaneous RDF as defined in Eq. 5.1. Eq. 5.10 becomes exact in the limit $\Delta r \to 0$. It is of course this expectation value (in MD approximated as a time average over a trajectory) that makes the link to statistical mechanics.

5.2.5 ∗Experimental determination of radial distribution function

Radial distribution functions, it turns out, can be observed by experiment. They can be determined form diffraction patterns of radiation with a wavelength comparable to interatomic distance. This means that for normal liquids with interatomic distances in the order of angstroms, we can use neutrons and X-rays, but not visible light. The quantity that is actually directly measured by diffraction experiments is the intensity $I(\theta)$ scattered in a direction at an angle θ of the incoming beam. If $\mathbf{k}_{in}$ and $\mathbf{k}_{out}$ are the wavevectors of the incoming respectively outgoing beam, the momentum transfer involved is

$$\mathbf{k} = \mathbf{k}_{out} - \mathbf{k}_{in} \tag{5.11}$$

where because the scattering is elastic $|\mathbf{k}_{out}| = |\mathbf{k}_{in}|$, and therefore

$$k = |\mathbf{k}| = \frac{4\pi}{\lambda_{in}} \sin(\theta/2) \tag{5.12}$$

To a very good approximation the observed scattered intensity can be separated into a atomic form factor $f(k)$ and structure factor $S(k)$

$$I(\theta) = f(k) N S(k) \tag{5.13}$$

The form factor is specific to the atomic species and also depends on instrumental corrections. The structure factor is given by

$$S(\mathbf{k}) = \frac{1}{N} \left\langle \sum_{l,m}^{N} \exp\left[i\mathbf{k} \cdot (\mathbf{r}_l - \mathbf{r}_m)\right] \right\rangle \tag{5.14}$$

and contains all the information on the position of the particles. Similar to the RDF we have used a more general formulation allowing for possible dependence on the direction of the momentum transfer as is the the case for example for Bragg scattering of crystals. For liquids, however the structure factor is isotropic and only depends on the magnitude $k = \mathbf{k}$ of the scattering vector. To relate the structure factor to the radial distribution we use the formal definition of the RDF in terms of Dirac delta functions (Eq. 5.5) and the Fourier transform representation of Dirac delta functions (see e.g. Riley, Hobson and Bench, *"Mathematical methods for Physics and engineering"*, section 13.1).

$$\delta(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} dk\, e^{ikx} \tag{5.15}$$

To proceed we first separate the sum in Eq. 5.14 in $l = m$ and $l \neq m$ terms

$$\begin{aligned} S(\mathbf{k}) =& \frac{1}{N}\left\langle \sum_{l}^{N} \exp\left[i\mathbf{k}\cdot\mathbf{0}\right]\right\rangle + \frac{1}{N}\left\langle \sum_{l\neq m}^{N} \exp\left[i\mathbf{k}\cdot(\mathbf{r}_l - \mathbf{r}_m)\right]\right\rangle \\ =& 1 + \frac{1}{N}\left\langle \sum_{l\neq m}^{N} \exp\left[i\mathbf{k}\cdot(\mathbf{r}_{lm})\right]\right\rangle \end{aligned} \tag{5.16}$$

and then evaluate the Fourier transform (in 3D) applying Eq. 5.14

$$\frac{1}{(2\pi)^3}\int d\mathbf{k}\, e^{i\mathbf{k}\cdot\mathbf{r}} S(\mathbf{k}) = \delta(\mathbf{r}) + \frac{1}{N}\left\langle \sum_{l\neq m}^{N} \delta(\mathbf{r}_{lm} - \mathbf{r})\right\rangle \tag{5.17}$$

Now we can substitute Eq. 5.5 to find

$$\frac{1}{(2\pi)^3}\int d\mathbf{k}\, e^{i\mathbf{k}\cdot\mathbf{r}} S(\mathbf{k}) = \delta(\mathbf{r}) + \frac{N}{V} g(\mathbf{r}) \tag{5.18}$$

or moving the delta function over to the l.h.s and absorbing it in the integral

$$\frac{1}{(2\pi)^3}\int d\mathbf{k}\, e^{i\mathbf{k}\cdot\mathbf{r}} \left[S(\mathbf{k}) - 1\right] = \frac{N}{V} g(\mathbf{r}) \tag{5.19}$$

The $g(r)$ can therefore be obtained from experiment by the (inverse) Fourier transform of the measured structure factor after subtracting the "selfcorrelation".

5.3 Pressure

In section 4.2.3 we showed how kinetic energy can be used to measure temperature in MD. The thermodynamic temperature is obtained as an average of the instantaneous temperature $\mathcal{T}$ of Eq. 4.48. The derivation of a suitable microscopic function $\mathcal{P}$ for the determination of instantaneous pressure is more involved. Most textbooks on statistical thermodynamics start from the relation of the thermodynamic pressure P to the Helmholtz free energy energy and, therefore, through the fundamental relation of Eq. 4.45, to the configurational partition function Z_N of Eq. (4.44).

$$P = -\left(\frac{\partial A}{\partial V}\right)_{N,T} = k_B T \left(\frac{\partial \ln Z_N}{\partial V}\right)_{N,T} \quad . \tag{5.20}$$

In order to perform the differentiation with respect to V, the volume dependence is eliminated from the real space integration boundaries in Eq. (4.44) by a scaling transformation

$$\mathbf{r}_i = L\mathbf{s}_i, \quad L = V^{\frac{1}{3}} \tag{5.21}$$

which yields the following expression for the derivative of Z_N

$$\frac{\partial Z_N}{\partial V} = \frac{\partial}{3L^2\partial L}\left(\int_0^1 L^3 d\mathbf{s}_1 \ldots \int_0^1 L^3 d\mathbf{s}_N \, \exp\left[-\frac{\mathcal{V}(L\mathbf{s}_1, \ldots L\mathbf{s}_N)}{k_B T}\right]\right) \tag{5.22}$$

Differentiating w.r.t to L and transforming back to unscaled coordinates we find

$$P = \frac{Nk_BT}{V} + \frac{1}{3V}\left\langle \sum_i^N \mathbf{r}_i \cdot \mathbf{f}_i \right\rangle \quad . \tag{5.23}$$

The first term in Eq. (5.23) arises from the L^3 factors of the Jacobian in Eq. 5.22 and represents an ideal gas contribution, independent of interatomic interactions. The effect of interactions is contained in the second term, which is proportional to the canonical expectation value of the total virial $\mathcal{W}$ of the forces on the atoms.

$$\mathcal{W} = \sum_i^N \mathbf{r}_i \cdot \mathbf{f}_i^{\text{int}} \quad . \tag{5.24}$$

The superscript "int" has been added to stress that only interatomic forces contribute to $\mathcal{W}$. Interaction with the walls of the container introduce an explicit dependence on volume in the potential which was not accounted for in our derivation of Eq. (5.23) and hence must be excluded from the virial pressure. For pair interactions (using Newtons third law Eq. 4.9) we can convert the virial of Eq. 5.24 in an explicitly translational invariant form

$$\mathcal{W} = \sum_{i,j>i}^N \mathbf{r}_{ij} \cdot \mathbf{f}_{ij} \quad , \tag{5.25}$$

where $\mathbf{f}_{ij}$ are the pair forces defined in Eq. (4.8).

The k_BT factor in the first (ideal gas) term in Eq. (5.23) can be interpreted an average over kinetic energy $\mathcal{K}$. This suggests adopting the phase function

$$\mathcal{P} = \frac{1}{3V}\left(2\mathcal{K} + \mathcal{W}\right) \tag{5.26}$$

as a measure for the instantaneous pressure. From a more intuitive point of view the above argument is somewhat unsatisfying: the interaction with the walls of the container, and hence also the coupling to the piston by which the pressure is applied, is completely ignored. Using the virial theorem of classical mechanics it is possible to give an alternative derivation of Eq. 5.26 which accounts for these forces and also gives a more convincing explanation of the contribution of kinetic energy to the instantaneous pressure (see e.g. Goldstein, *Classical Mechanics* pag. 84).

Unfortunately, periodic boundary conditions as applied in MD introduce a nasty complication in the computation of pressure. Because of the effective volume dependence of the forces through the distance between images, Eq. (5.24) cannot be used for the determination of pressure. The correct procedure to compute the pressure in periodic systems is to use the volume derivative of the full lattice sum for the interaction energy (Eq. 4.56) as the force virial. This is usually a painful derivation. However, for short range pair interactions treated by a spherical cutoff smaller than half the length of the MD cell, the system serving as our model for a fluid, the alternative expression Eq. 5.25 saves us this trouble: It is already in the correct form, because, with all interactions beyond a half period neglected (set to zero), there is no dependence on volume.

5.4 Appendix 2: Code samples

5.4.1 ∗Sampling the radial distribution function

In MD or MC codes radial distribution functions are estimated by making a histogram of particle distances. The accumulation of such a histogram could be part of the task of a *sample* routine at the end of a MD or MC step. However, since the particle distances are already determined in the *force* routine in MD (or the *energy* routine for MC) for the computation of energies and forces, the binning of distances is usually folded into the force(energy) loop for reasons of computational efficiency. The required extension is only two or three lines of code as is illustrated below for the force loop of Code 4.29.

```
do i := 1, natom − 1
      do j := i + 1, natom
            call pbc (l, x(i), y(i), z(i), x(j), y(j), z(j), dx, dy, dz)
            dr := sqrt (dx ∗ ∗2 + dy ∗ ∗2 + dz ∗ ∗2)
            ig := int (dr/delr) + 1
            if (ig ≤ nbin) g(ig) := g(ig) + 2
            if (dr < rc) then
              · · ·                    \∗ calculations of forces and energies, see Code 4.27 ∗\
            endif
      enddo
enddo
```

Code 5.1: Force loop with binning of particle distances for radial distribution function

The histogram is accumulated in an array g, the elements of which are the bins. To enter the distance of a pair of atoms in the appropriate bin, the minimum image distance, as determined by the application of the periodic boundary conditions procedure, is converted to a positive integer ig. if ig is not larger than $nbin$, the maximum number of bins in the histogram, the counter in the bin ig is increased by 2 (accounting for the equivalent (i, j) and (j, i) particle pairs). The parameter $delr$ is the bin width. Its value is set elsewhere at the beginning of the program. The initialization routine const_md (Code 4.32) is a suitable place for this. This can be done by adding the lines

```
subroutine const_md (natom, clen, · · · , lbox, · · · nbin, delr, g, · · · )
  ⋮
lbox := clen ∗ lbox
delr := lbox/ (2 ∗ nbin)          \∗ bin width, nbin is maximum number of bins ∗\
do n := 1, nbin                                                  \∗ zeroing bins ∗\
      g(i) := 0
enddo
  ⋮
end
```

Code 5.2: Extension of Code 4.32 for the preparation for the binning of distances

The binning interval, i.e the maximum distance in the histogram, is set to half the box length. After completion of the last iteration step, having accumulated results over $nsteps$, the contents of the histogram g is normalized and transformed to a radial distribution function according to the definition of Eq. 5.1. We will also give to code for this operation.

```
pi := 3.14159
rho := natom/lbox**3                                   \* density *\
rm := 0
zm := 0
do n := 1, nbin
      r := delr * i                                    \* maximum distance in bin i *\
      r(i) := r - 0.5 * delr                           \* average distance in bin i *\
      g(i) := g(i)/ (natom * nstep)                    \* number of particles in shell i *\
      z(i) := zm + g(i)                                \* coordination number up to end of bin i *\
      vb := (4/3) * pi * (r**3 - rm**3)                \* volume of shell i *\
      g(i) := g(i)/ (vb * rho)                         \* radial distribution function for distance i *\
      rm := r
      zm := z(i)
enddo
```

Code 5.3: Converting the distance histogram to a radial distribution function

The piece of Code 5.3 produces two arrays g and z to be tabulated, or better plotted, as function of distances $r(i)$. g is the radial distribution function $g(r)$ and z the running coordination number $n_c(r)$, i.e the integral of Eq.5.4 up to distance r.

Project D: *Determine the radial distribution function of the liquid argon system of Project C. How long a trajectory is needed for reasonable convergence of the g of r?*

chapterMeasuring dynamical properties

In this chapter we will address the question how we can extract information on the dynamics of a system from a MD trajectory. Clearly watching an animation of the motion of the particles on a graphics workstation will give us a very good idea what is going on. However, for a comparison to experimental observation or predictions of analytical theory we need a well defined statistical procedure to characterize microscopic dynamics. Time correlation functions provide such a statistical tool. Moreover, many spectroscopic and transport quantities measured by experiment can be expressed in terms of time correlation functions (or their Fourier transforms).

5.5 Velocity autocorrelation function

The velocity autocorrelation function is perhaps the best known example of a time correlation function. It is a very convenient probe of particle dynamics. The velocity autocorrelation function is used for the interpretation (assignment) of vibrational spectra of liquids and is also connected with the phenomenon of diffusion. The velocity autocorrelation function of an N atom system is generally defined as

$$c_{vv}(\tau) = \left(\sum_{i=1}^{N} \langle \mathbf{v}_i^2 \rangle \right)^{-1} \sum_{i=1}^{N} \langle \mathbf{v}_i(\tau) \cdot \mathbf{v}_i(0) \rangle \tag{5.27}$$

where $\mathbf{v}_i(t)$ is the velocity vector of atom i at time t. The scalar product $\mathbf{v}_i(\tau) \cdot \mathbf{v}_i(0) = v_x(\tau)v_x(0) + v_y(\tau)v_y(0) + v_z(\tau)v_z(0)$ compares the velocity at time $t = \tau$ with the velocity at the initial time $t = 0$. The angular brackets denote a statistical equilibrium average. What

this means in the case of observables sampled at different times will be explained in the next section. The prefactor is a normalization constant and, for velocities, is easily evaluated using equipartition

$$\sum_{i=1}^{N}\langle \mathbf{v}_i(0)\cdot\mathbf{v}_i(0)\rangle = \sum_{i=1}^{N}\langle \mathbf{v}_i^2\rangle = \frac{3Nk_BT}{m} \tag{5.28}$$

where we have assumed that all particles have equal mass $m_i = m$. Again this will become more clear after the formal introduction of time correlation in the next section.

Fig. 5.5 shows an example of the velocity auto correlation function of liquid argon. The short time and long time behavior of $c_{vv}(\tau)$ reveal the two different faces of liquids which also showed in the radial distribution function: On a short time (distance) scale, liquids are similar to solids. Relaxation typical of liquids becomes manifest at longer times.

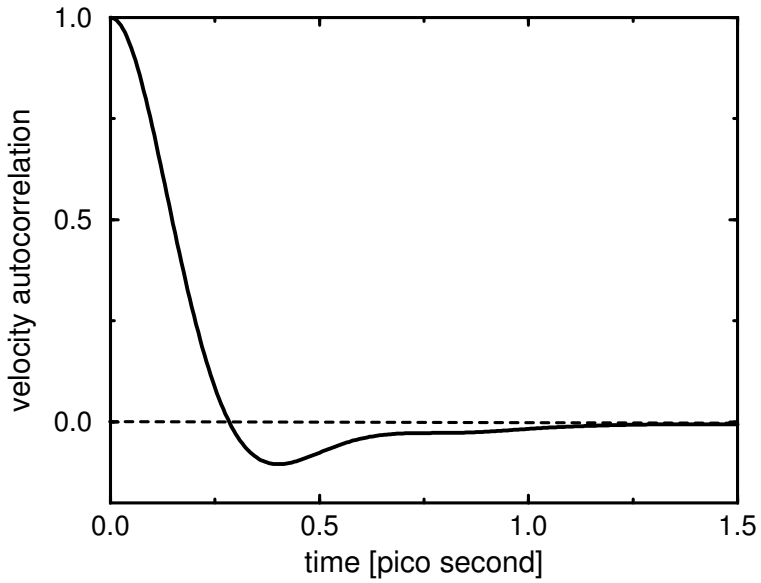


Figure 5.5: Velocity autocorrelation function for liquid argon. The minimum at ≈ 0.4ps reflects the oscillation of an atom in the potential well due to interaction with its neighbors (see section 5.7.1). This oscillation, however is strongly dampened due to relaxation effects in the liquid. The exponential tail can be related to diffusive motion (see section 5.8.2).

5.6 Time correlation functions

5.6.1 Comparing properties at different times

Time correlation functions measure statistical correlations across time signals (see figure 5.6). So, given a signal, for example voltage noise in an electrical device or a trajectory in MD, the value of some observable $B(t)$ at a certain time t is compared to the value of observable $A(t+\tau)$ a time τ later by multiplying these values. This is illustrated in Fig. 5.6 for two instants (t and t') along a trajectory. This product varies with the reference time t. What we are interested in is what happens "on average" over a period τ. So, averaging over reference times t we define

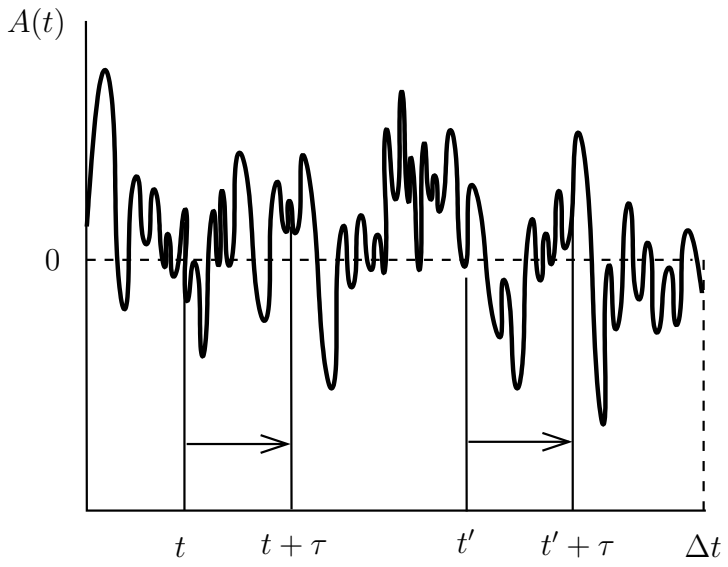


Figure 5.6: Schematic representation of an observable fluctuating in time ("signal") as recorded in a MD simulation of duration Δt. The correlation over a length of time τ is sampled at two (arbitrary) instants t and t'. The time correlation function (Fig. 5.7) is a reference over these reference times.

the time correlation as the integral

$$C_{AB}(\tau) = \frac{1}{\Delta t - \tau} \int_0^{\Delta t - \tau} dt\, A(t+\tau) B(t) \tag{5.29}$$

where Δt is the length of the trajectory. Of course we can not look beyond the end of the signal. The stretch of time available for averaging for given τ is, therefore, not the full length of the signal Δt but $\Delta t - \tau$, hence the time τ is subtracted from Δt in Eq. 5.29 accordingly. Fig. 5.7 is an example of a typical time correlation function as derived from stationary time signals such as shown in Fig. 5.6.

As explained in section 4.2.1 time averages in molecular dynamics are sampled at discrete time intervals. The minimum length of such an interval is the time step δt (see Eq. 4.36). The separation in time τ in Eq. 5.29 is necessarily an integer multiple $\tau = m\delta t$ of the time step where m must be smaller (or equal) to the total number of time steps M. Assuming the signal is sampled every time step, the MD estimator of the time correlation function Eq. 5.29 is therefore an average over $M - m$ time steps

$$C_{AB}(\tau) \approx \overline{C}_{AB}(m\delta t) = \frac{1}{M-m} \sum_{n=1}^{M-m} A(t_{n+m})\, B(t_n) \tag{5.30}$$

Notice that the number of samples, and therefore the statistical accuracy, decreases when τ (i.e m) increases leaving only just one data point for $m = M - 1$.

5.6.2 ∗Ensemble averages and time correlations

Eq. 5.29 and 5.30 are operational definitions of time correlation functions: this is how they are measured. In this respect Eq. 5.30 can be compared to the definition of Eq. 5.1 of the

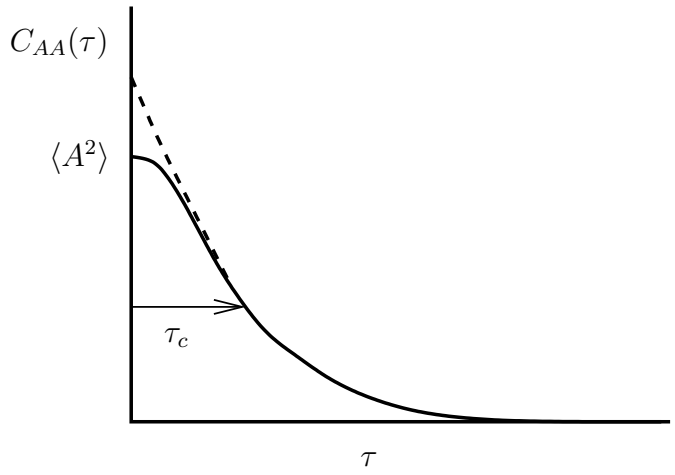


Figure 5.7: Time autocorrelation function as defined in Eq. 5.29 corresponding to the signal of figure 5.6 (schematic, solid curve with $\langle A \rangle = 0$). The correlation is approximately exponential in time expect at very short times, where inertial effects dominate. The dashed curve indicates the long time exponential decay extrapolated to short times. (see section 5.6.2).

radial distribution function (RDF), measuring (simultaneous) correlations in space in terms of a histogram of pair distances. What remains to be done is to relate these definitions to the ensemble distribution functions of statistical mechanics, as we did in section 5.2.4 for the RDF.

In order to establish the link to statistical mechanics we will return to the formal treatment of the time evolution $A(t)$ of an observable $\mathcal{A}\left(\mathbf{r}^N, \mathbf{p}^N\right)$ of Eq. 4.33. We will be more precise about the dependence on the initial condition $\left(\mathbf{r}_0^N, \mathbf{p}_0^N\right)$ and add the phase space point at $t = 0$ as an explicit argument in the definition of $A(t)$ as we did already in Eq. 4.18 for the trajectory (see also Fig. 5.8)

$$A(t) \equiv \mathcal{A}\left(\mathbf{r}^N(t), \mathbf{p}^N(t)\,;\, \mathbf{r}_0^N, \mathbf{p}_0^N\right) \tag{5.31}$$

According to Eq. 5.29 the time correlation function of the quantities A and B is a product of the values of A and B at two different points in time along the same trajectory, hence setting $t = 0$

$$A(\tau)B(0) = \mathcal{A}\left(\mathbf{r}^N(\tau), \mathbf{p}^N(\tau)\,;\, \mathbf{r}'^N, \mathbf{p}'^N\right) \mathcal{B}\left(\mathbf{r}'^N, \mathbf{p}'^N\right)$$

with coordinates $\mathbf{r}'^N, \mathbf{p}'^N$ playing the role of "variable" initial conditions, each set $\mathbf{r}'^N, \mathbf{p}'^N$ giving a different trajectory. Under equilibrium conditions the probability (density) for finding the system at a phase point $\mathbf{r}'^N, \mathbf{p}'^N$ is given by the ensemble distribution function ρ_{eq} of Eq. 4.39 (or in case of MD runs Eq. 4.38). This suggests that we can average over all the different trajectories using ρ_{eq} as weight

$$C\left(\tau\right)_{AB} \equiv \langle A(\tau)B(0) \rangle = \int d\mathbf{r}'^N d\mathbf{p}'^N \rho_{eq}\left(\mathbf{r}'^N, \mathbf{p}'^N\right) \mathcal{A}\left(\mathbf{r}^N(\tau), \mathbf{p}^N(\tau)\,;\, \mathbf{r}'^N, \mathbf{p}'^N\right) \mathcal{B}\left(\mathbf{r}'^N, \mathbf{p}'^N\right) \tag{5.32}$$

The angular brackets in Eq. 5.32 have exactly the same meaning as in Eq. 4.37 or 4.40, namely the expectation value in an equilibrium ensemble. Similarly the first equality of Eq. 4.37 applies stating that if the system is ergodic (chaotic) the time average over a long trajectory should converge to the appropriate microcanonical expectation value, i.e over the hyper surface at constant energy E. Consequently we can use the time correlation obtained from an average

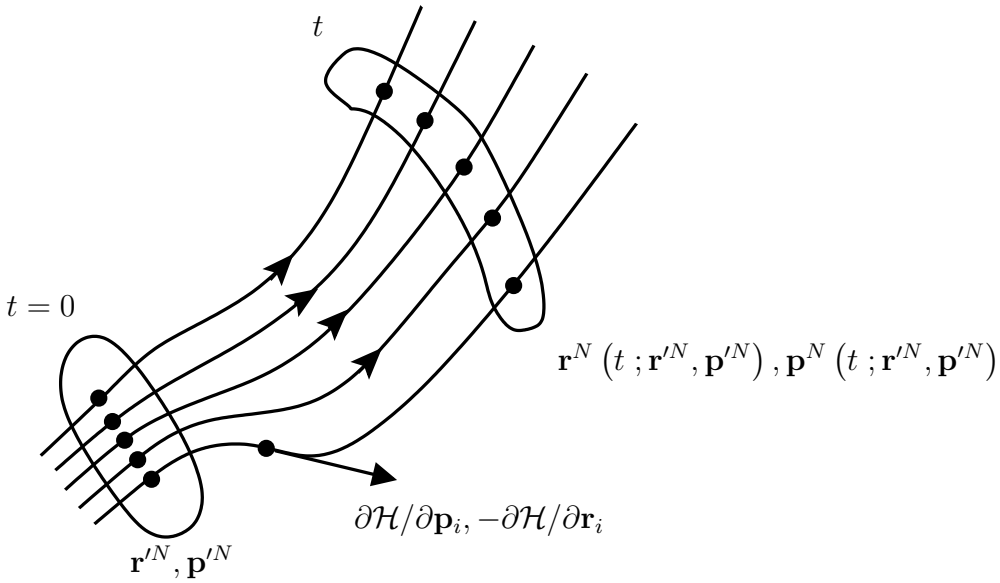


Figure 5.8: Bundle of adjacent trajectories in phase space with an initial phase point at $t = 0$ and a phase point on the same trajectory at time t. The arrow indicates the tangent to the trajectory according to Hamilton's equation of motion, Eq. 5.33, 5.34. A trajectory is completely determined by any of its point, hence there are no intersections.

over a sufficiently long MD trajectory (Eq. 5.29 or 5.30) as an approximation to the statistical time correlation function of Eq. 5.32.

5.6.3 ∗Dynamics in phase space

The dynamics in equilibrium systems consists of stationary motion and as a result the corresponding time correlation functions (Eq. 5.32) have a number of fundamental symmetry properties. As discussed in section 4.2.2, an equilibrium ensemble distribution functions $\rho_{eq}\left(\mathbf{r}^N,\mathbf{p}^N\right)$ can be written as functions of the Hamiltonian (Eq. 4.14). However, not only the equilibrium statistics but also the dynamics in phase space is controlled by the Hamiltonian. To see this we will first derive an equation of motion for the position variables $\mathbf{r}^N(t)$ and momentum variables $\mathbf{p}^N(t)$ describing a trajectory in phase space. This means, in the true spirit of the phasespace representation, that the momentum $\mathbf{p}_i$ of a particle is now considered as an independent dynamical degree of freedom with its own equation of motion. The phase space equations of motion are obtained by separating Newton's equation Eq. 4.3, which is a second order differential equation in time, in two first order equations called Hamilton's equations

$$\dot{\mathbf{r}}_i = \frac{\partial\mathcal{H}}{\partial\mathbf{p}_i} \tag{5.33}$$

$$\dot{\mathbf{p}}_i = -\frac{\partial\mathcal{H}}{\partial\mathbf{r}_i} \tag{5.34}$$

These equations can easily be verified by substitution. Differentiating the Hamiltonian to momentum as required by Eq. 5.33

$$\frac{\partial\mathcal{H}}{\partial\mathbf{p}_i} = \frac{\mathbf{p}_i}{m_i} = \dot{\mathbf{r}}_i$$

we recover the definition of linear momentum. Similarly evaluating the derivative w.r.t to position

$$\frac{\partial \mathcal{H}}{\partial \mathbf{r}_i} = \frac{\partial \mathcal{V}}{\partial \mathbf{r}_i} = -\mathbf{f}_i$$

and we see that Eq. 5.34 is equivalent to Newton's equation of motion. Hamilton's equations, however, are more general than this and for example also hold for polar spherical coordinates r, θ, ϕ with appropriate definitions of momentum p_r, p_θ, p_ϕ.

While in principle equivalent to Newton's equations of motion, the advantage of the Hamiltonian formalism is that it is very convenient for expressing general relations valid for the dynamics of functions defined in phase space. For example, if we substitute Hamilton's equations Eq. 5.33, 5.34 in the expression Eq. 4.34 for the total time derivative function of $\mathcal{A}\left(\mathbf{r}^N, \mathbf{p}^N\right)$

$$\frac{d\mathcal{A}}{dt} = \sum_{j=1}^{N} \left[\dot{\mathbf{r}}_j \cdot \frac{\partial \mathcal{A}}{\partial \mathbf{r}_j} + \dot{\mathbf{p}}_j \cdot \frac{\partial \mathcal{A}}{\partial \mathbf{p}_j} \right] \tag{5.35}$$

we obtain

$$\frac{d\mathcal{A}}{dt} = \sum_{j=1}^{N} \left[\frac{\partial \mathcal{H}}{\partial \mathbf{p}_j} \cdot \frac{\partial \mathcal{A}}{\partial \mathbf{r}_j} - \frac{\partial \mathcal{H}}{\partial \mathbf{r}_j} \cdot \frac{\partial \mathcal{A}}{\partial \mathbf{p}_j} \right] \tag{5.36}$$

Taking for $\mathcal{A}$ the Hamiltonian $\mathcal{H}$ we find, of course, $d/dt\mathcal{H} = 0$. Hence, in this representation conservation of energy is a trivial matter of interchanging factors in a product. A phase function of special interest is the equilibrium ensemble distribution function $\rho_{eq}\left(\mathbf{r}^N, \mathbf{p}^N\right)$ of Eq. 4.39. ρ_{eq} is a function of the total energy $\mathcal{H}$, which is a constant of motion (see Eq. 4.17). As a result, ρ_{eq} is a constant of motion as well.

$$\frac{d\rho_{eq}}{dt} = \frac{\partial \rho_{eq}}{\partial \mathcal{H}} \frac{d\mathcal{H}}{dt} = \frac{\partial \rho_{eq}}{\partial \mathcal{H}} \sum_{j=1}^{n} \left[\dot{\mathbf{r}}_j \cdot \frac{\partial \mathcal{H}}{\partial \mathbf{r}_j} + \dot{\mathbf{p}}_j \cdot \frac{\partial \mathcal{H}}{\partial \mathbf{p}_j} \right] = 0 \tag{5.37}$$

The same is true for the microcanonical probability distribution of Eq. 4.38, since also the Dirac delta function $\delta\left(\mathcal{H}\left(\mathbf{r}^N, \mathbf{p}^N\right) - E\right)$ can be treated as a function of $\mathcal{H}$.

5.6.4 ∗Symmetries of equilibrium time correlations

With a vanishing total time derivative the ensemble is invariant in time, i.e. the ensemble distribution function is conserved along a trajectory

$$\rho_{eq}\left(\mathbf{r}^N(t), \mathbf{p}^N(t)\,;\, \mathbf{r}'^N, \mathbf{p}'^N\right) = \rho_{eq}\left(\mathbf{r}'^N, \mathbf{p}'^N\right) \tag{5.38}$$

This seems an obvious and necessary feature for an equilibrium ensemble. However this has a number of important implications for time correlation functions: The first is that the reference point in time (t in Eq. 5.29) is irrelevant

$$\langle A(t+\tau)B(t)\rangle = \langle A(\tau)B(0)\rangle \tag{5.39}$$

To prove Eq. 5.39 we write out $\langle A(t+\tau)B(t)\rangle$ according to Eq. 5.32

$$\begin{aligned}\langle A(t+\tau)B(t)\rangle = \int d\mathbf{r}'^N d\mathbf{p}'^N \rho_{eq}\left(\mathbf{r}'^N, \mathbf{p}'^N\right) \times \\ \mathcal{A}\left(\mathbf{r}^N(t+\tau), \mathbf{p}^N(t+\tau)\,;\, \mathbf{r}'^N, \mathbf{p}'^N\right) \mathcal{B}\left(\mathbf{r}^N(t), \mathbf{p}^N(t)\,;\, \mathbf{r}'^N, \mathbf{p}'^N\right)\end{aligned} \tag{5.40}$$

and substitute

$$\mathcal{A}\left(\mathbf{r}^N(t+\tau), \mathbf{p}^N(t+\tau)\,;\,\mathbf{r}'^N, \mathbf{p}'^N\right) = \\ \mathcal{A}\left(\mathbf{r}^N(\tau), \mathbf{p}^N(\tau)\,;\,\mathbf{r}^N\left(t\,;\,\mathbf{r}'^N, \mathbf{p}'^N\right), \mathbf{p}^N\left(t\,;\,\mathbf{r}'^N, \mathbf{p}'^N\right)\right) \tag{5.41}$$

and a similar expression for B with $\tau = 0$. Eq. 5.41 is a complicated way of saying that trajectories in phase space are unique (Fig. 5.8). We can take any point between t and $t = 0$ as an initial condition and will end up in the same state. Substituting Eq. 5.38, we can change the integration variables in Eq. 5.40 to the phase space coordinates at time t

$$d\mathbf{r}'^N d\mathbf{p}'^N = d\mathbf{r}^N\left(t\,;\,\mathbf{r}'^N, \mathbf{p}'^N\right) d\mathbf{p}^N\left(t\,;\,\mathbf{r}'^N, \mathbf{p}'^N\right)$$

and we find Eq. 5.39 (Strictly speaking, the change of integration variables needs justification. It is allowed because volume in phase space is conserved, i.e. the flow in Fig. 5.8 is incompressible.)

Eq. 5.39 leads to a number of further convenient symmetry properties. For example displacing the origin of time in a autocorrelation function (observable A correlated with its value at different times) by $-\tau$ gives

$$\langle A(\tau)A(0)\rangle = \langle A(0)A(-\tau)\rangle$$

Then interchanging the two factors (allowed for classical quantities but not for quantum operators!) we find that autocorrelation functions are symmetric in time

$$\langle A(0)A(-\tau)\rangle = \langle A(0)A(\tau)\rangle \tag{5.42}$$

Another useful relation concerns the correlations involving time derivatives (velocities). Time derivatives can be "exchanged" between observables according to

$$\left\langle \dot{A}(0)B(\tau)\right\rangle = -\left\langle A(0)\dot{B}(\tau)\right\rangle \tag{5.43}$$

Eq. 5.43 follows because according to Eq. 5.39 time correlations are independent of the reference time t and therefore

$$\frac{d}{dt}\langle A(t+\tau)B(t)\rangle = 0$$

Carrying out the differentiation using the product rule we find

$$\langle \dot{A}(t+\tau)B(t)\rangle = -\langle A(t+\tau)\dot{B}(t)\rangle$$

Setting $t = 0$ gives the relation of Eq. 5.43. Time translation symmetry (an not time inversion symmetry) is the reason that static correlations between a quantity and its velocity vanish. In the limit of zero correlation time τ, the static (instantaneous) correlation is obtained. Setting in Eq. 5.43 τ to zero for $B(t) = A(t)$

$$\left\langle \dot{A}(0)A(0)\right\rangle = -\left\langle A(0)\dot{A}(0)\right\rangle = -\left\langle \dot{A}(0)A(0)\right\rangle$$

and hence

$$\left\langle \dot{A}(0)A(0)\right\rangle = 0 \tag{5.44}$$

In equilibrium ensembles, the cross correlation between any observable and its velocity is zero.

5.6.5 Fluctuations and correlation times

The fluctuations of a quantity in equilibrium, for example the velocity of a particle or the kinetic energy, may be noisy but the time dependence is stationery: the signal, although it may never repeat itself in detail, looks essentially the same all the time (Fig. 5.6). The time scale of the fluctuations is characteristic for both the observable and the state of the system and can often be estimated from experimental data. One of the key quantities describing fluctuations is their correlation or "life time" time τ_{cor}. For times longer than τ_{cor} the motion of quantities A and B has lost correlation, i.e $B(0)$ and $A(\tau)$ become statistically independent variables. As a result the corresponding time correlation Eq. 5.32 can be factorized

$$C_{AB}(\tau \gg \tau_{cor}) = \langle A \rangle \langle B \rangle \tag{5.45}$$

The correlation at zero time is non other than the equilibrium average of a product of observables and also contains no dynamical information. Hence, for the study of relaxation of the fluctuations of a quantity A it is convenient to introduce the normalized time auto correlation function

$$\begin{aligned} c_{AA}(\tau) &= \left\langle \delta A^2 \right\rangle^{-1} \langle \delta A(\tau) \delta A(0) \rangle \\ \delta A(t) &= A(t) - \langle A \rangle \end{aligned} \tag{5.46}$$

which starts out with unit value at $\tau = 0$ and decays to zero for $\tau \to \infty$. A crude first approximation describing this behavior is an exponent (Fig. 5.7)

$$c_{AA}(\tau) = \exp\left[-\tau/\tau_{cor}\right] \tag{5.47}$$

If decay in time were exactly exponential we have

$$\tau_{cor} = \int_0^\infty d\tau \, c_{AA}(\tau) \tag{5.48}$$

Also for correlation functions with more structure than an exponent (see Fig. 5.7 and section 5.7) the integral Eq. 5.48 can be used as a measure for relaxation time. Note that on a short time scale the time dependence of auto correlation functions must deviate from an exponent because of incompatibility with Eq. 5.42 forcing the derivative w.r.t τ at $\tau = 0$ to vanish:

$$\lim_{\tau \to 0} \frac{d}{d\tau} c_{AA}(\tau) = 0$$

the derivative of an exponent of Eq. 5.47, on the other hand, remains finite

$$\lim_{\tau \to 0} \frac{d}{d\tau} \exp\left[-\tau/\tau_{cor}\right] = -\frac{1}{\tau_{corr}} < 0$$

(see also next section).

5.7 Velocity autocorrelation and vibrational motion

5.7.1 Short time behavior

The short timer behavior of the velocity autocorrelation function (the "wiggle" in Fig. 5.5) contains information about the vibrational motion in the system. One way to see this is to

investigate the Taylor expansion of $c_{vv}(\tau)$ up to second order in τ

$$\langle \mathbf{v}_i(\tau)\mathbf{v}_i(0)\rangle = \langle \mathbf{v}_i\mathbf{v}_i\rangle + \langle \dot{\mathbf{v}}_i\mathbf{v}_i\rangle\tau + \langle \ddot{\mathbf{v}}_i\mathbf{v}_i\rangle\frac{\tau^2}{2} + \cdots \tag{5.49}$$

The zero order term will give unity after dividing by the normalization constant (see Eq. 5.27). The first order term vanishes because of the symmetry relation Eq. 5.44 applied to the velocity

$$\langle \dot{\mathbf{v}}_i\mathbf{v}_i\rangle = 0 \tag{5.50}$$

For an interpretation of the second order term we first apply partial integration (chain rule)

$$\langle \ddot{\mathbf{v}}_i\mathbf{v}_i\rangle = \langle \dot{\mathbf{v}}_i\mathbf{v}_i\rangle - \langle \dot{\mathbf{v}}_i^2\rangle \tag{5.51}$$

The first term vanishes again . Substuting the equation of motion in the second find

$$\langle \ddot{\mathbf{v}}_i\mathbf{v}_i\rangle = -\frac{1}{m_i^2}\langle \mathbf{f}_i^2\rangle \tag{5.52}$$

The average squared force can be related to the local curvature of the potential using the generalized equipartition relation

$$\left\langle \mathcal{A}\frac{\partial\mathcal{H}}{\partial\xi}\right\rangle = k_BT\left\langle \frac{\partial\mathcal{A}}{\partial\xi}\right\rangle \tag{5.53}$$

where ξ is either a component of the coordinate vector $\mathbf{r}_i$ or the momentum vector $\mathbf{p}_i$. Eq. 5.53 can be easily derived from the expression Eq. 4.40 by partial integration. For example for $\xi = \mathbf{r}_i$

$$\begin{aligned}\left\langle \mathcal{A}\frac{\partial\mathcal{H}}{\partial\mathbf{r}_i}\right\rangle =& \frac{f(N)}{Q_N}\int d\mathbf{r}^N d\mathbf{p}^N \mathcal{A}\left(\mathbf{r}^N,\mathbf{p}^N\right)\left(\nabla_{\mathbf{r}_i}\mathcal{H}\left(\mathbf{r}^N,\mathbf{p}^N\right)\right)\exp\left[-\beta\mathcal{H}\left(\mathbf{r}^N,\mathbf{p}^N\right)\right]\\ =& -k_BT\frac{f(N)}{Q_N}\int d\mathbf{r}^N d\mathbf{p}^N \mathcal{A}\left(\mathbf{r}^N,\mathbf{p}^N\right)\nabla_{\mathbf{r}_i}\exp\left[-\beta\mathcal{H}\left(\mathbf{r}^N,\mathbf{p}^N\right)\right]\\ =& k_BT\frac{f(N)}{Q_N}\int d\mathbf{r}^N d\mathbf{p}^N \left(\nabla_{\mathbf{r}_i}\mathcal{A}\left(\mathbf{r}^N,\mathbf{p}^N\right)\right)\exp\left[-\beta\mathcal{H}\left(\mathbf{r}^N,\mathbf{p}^N\right)\right] = k_BT\left\langle\frac{\partial\mathcal{A}}{\partial\mathbf{r}_i}\right\rangle\end{aligned}$$

Eq. 5.53 is very useful for the evaluation of variances. Applying Eq. 5.53 the force variance term in Eq. 5.52 can be transformed to

$$\omega_i^2 = \frac{\langle \dot{\mathbf{v}}_i^2\rangle}{\langle \mathbf{v}_i^2\rangle} = \frac{1}{3m_i}\left\langle\frac{\partial^2V(\mathbf{r})}{\partial\mathbf{r}_i^2}\right\rangle \tag{5.54}$$

and, hence, can be interpreted as an effective harmonic frequency. Inserting in Eq. 5.49 we obtain the quadratic short time approximation

$$c_{vv}(\tau) = \left(1 - \frac{1}{2}\omega_i^2\tau^2 + ...\right) \tag{5.55}$$

Now we can understand the wiggle in the velocity autocorrelation in Fig. 5.5 as the result of a strongly dampened oscillation of an atom in the potential due to the fluctuating coordination shell of nearest neighbors, i.e. the same structure that was also responsible for the first peak in the radial distribution of Fig. 5.1.

5.7.2 Vibrational dynamics in molecular liquids

The bonds holding molecules together are much stronger than the interactions between molecules controlling the structure and dynamics of liquids. We therefore expect the frequencies of molecular vibrations such as bond stretching and bond bending, to be considerable higher than the frequencies characterizing the severely dampened harmonic motion in the average potential established by intermolecular interactions as discussed in section 5.7.1. In the velocity autocorrelation function of molecular liquids, this effective separation in time scales of intra and intermolecula g over a sizable numb 5.9.

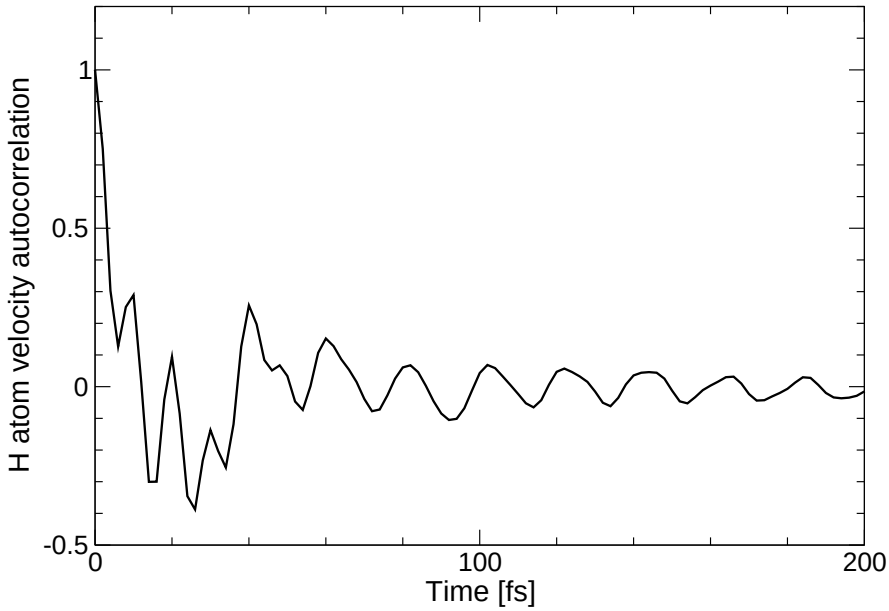


Figure 5.9: Velocity autocorrelation function of the H atoms in liquid water showing the multi-frequency oscillations related to the vibrational modes of H_2O molecules. As the period of these intra-molecular oscillations is considerably shorter than the time scale of relaxation processes due to intermolecular interactions (hydrogen bonding), the velocity autocorrelation exhibits a relatively large number of these oscillations before relaxing to zero (compare Fig. 5.5)
.

While containing a wealth of information on the dynamics in the liquid, this information is not easy to interpret or quantify by looking at the time dependence of the velocity autocorrelation. More revealing is its Fourier transform (spectrum) defined as:

$$f(\omega) = \int_{-\infty}^{\infty} d\tau \, e^{-i\omega\tau} c_{vv}(\tau) \tag{5.56}$$

where the correlation at negative times is obtained from the symmetry relation Eq. 5.42

$$c_{vv}(-\tau) = c_{vv}(\tau) \tag{5.57}$$

In virtue of Eq. 5.57, the spectrum could be equally obtained from a cosine transform using only positive time axis.

$$f(\omega) = 2 \int_{0}^{\infty} d\tau \cos(\omega\tau) \, c_{vv}(\tau) \tag{5.58}$$

and hence also the spectrum is symmetric in ω

$$f(-\omega) = f(\omega) \tag{5.59}$$

The inversion theorem for Fourier transforms allows us to express the time correlation function as

$$c_{vv}(\tau) = \frac{1}{2\pi} \int_{-\infty}^{\infty} d\omega \, e^{i\omega\tau} f(\omega) \tag{5.60}$$

where the spectrum at negative frequency is given by Eq. 5.59. Eq. 5.60 is often referred to as the spectral resolution of the velocity autocorrelation. Fig. 5.10 shows the spectrum of the velocity autocorrelation of liquid argon of Fig. 5.5.

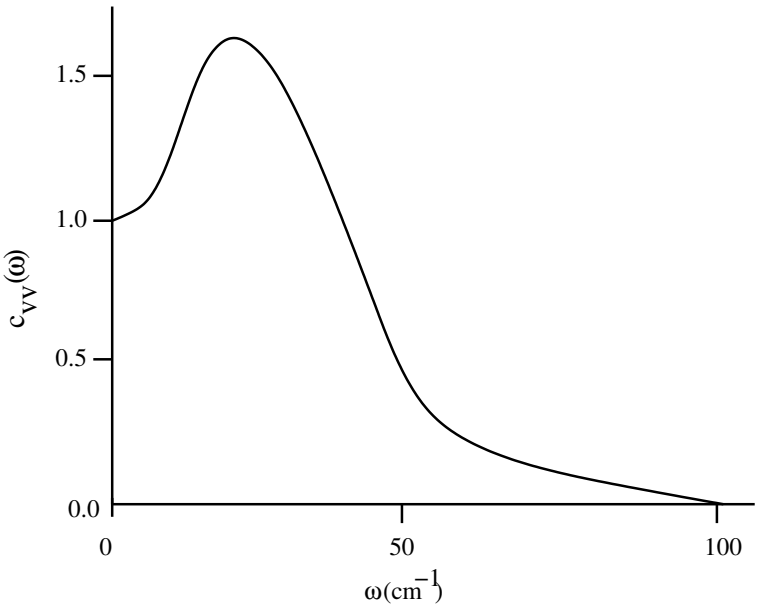


Figure 5.10: Fourier transform (spectrum) of the velocity autocorrelation of liquid argon of Fig. 5.5. The peak corresponds to the frequency of the dampened oscillation of an argon atom of a cage formed by its nearest neighbors. The width of the peak is inversely propertional to the relaxation time.

There is only one peak in Fig. 5.10 as there is indeed only one "wiggle" in Figure 5.5. In constrast, Figure 5.11, giving the spectrum obtained according to Eq. 5.56 of the H-atom velocity autocorrelation for liquid water of Fig. 5.9, shows several peaks. The OH stretch and HOH bending motion now clearly stand out as two well resolved bands at the high end of the frequency spectrum. The peaks appear at approximately the same frequency as for a gas-phase molecule. The effect of the intermolecular interactions is largely manifested in a broadening of these peaks (note that the stretching motion is much more affected compared to the bending). The broad band at low frequency ($< 1000\text{cm}^{-1}$) is absent for single molecules in vacuum and is the result of vibrational motion of molecules relative to each other. For the H atom spectrum of liquid water this motion is dominated by hydrogen bond bending, also called libration.

5.7.3 ∗Time correlation functions and vibrational spectroscopy

Similar to the radial distribution function (section 5.2.5) also time correlation functions are (more or less) directly accessible to experiment in Fourier transformed representation. For the

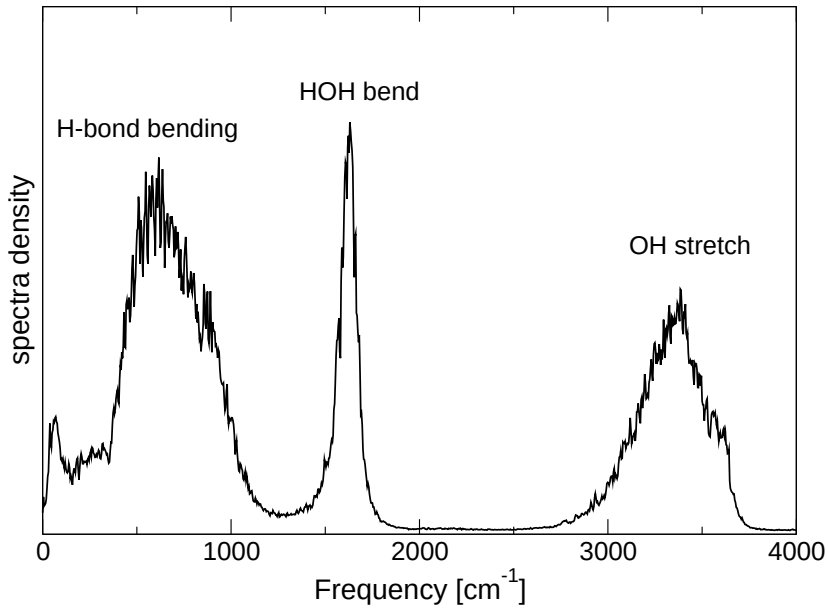


Figure 5.11: Fourier transform (spectrum) of the H-atom velocity autocorrelation of liquid water of Fig. 5.9. The stretching and intra molecular bending motion motion show up as two well resolved peaks at approximately the frequency of a gas-phase molecule. The low frequency part of the spectrum (¡ 1000 cm^{-1}) is the result of the strong intermolecular interactions (hydrogen bonding) in liquid water.

radial distribution function the key quantity was the structure factor as measured in diffraction experiments. The dynamics of a system is probed by the absorption of monochromatic infra red light. Raman spectra provide similar information. We will focus here on infrared spectroscopy.

As explained in the lectures on molecular spectroscopy, the absorption of single molecules is determined by the interaction of the molecular dipole moment with the oscillating electric field of the light source (laser). In fact the absorption is proportional to the square of the transition dipole. The absorption $\alpha(\omega)$ at frequency ω by a condensed molecular system such as a molecular liquid (or solvent) can, in first approximation, be described by a classical counter part of a squared transition dipole, which turns out to be the Fourier transform of the dipole time correlation function,

$$\alpha(\omega) = g(\omega) I(\omega) \tag{5.61}$$

where the intensity $I(\omega)$ is given by

$$I(\omega) = \int_{-\infty}^{\infty} d\tau \, e^{-i\omega\tau} \, \langle \mathbf{M}(\tau) \cdot \mathbf{M}(0) \rangle \tag{5.62}$$

The quantity $\mathbf{M}$ is the total dipole moment per unit volume (polarization) of the sample. The total dipole moment is the sum of all the molecular dipoles $\mathbf{d}_i$ and thus

$$\mathbf{M} = \frac{1}{V} \sum_{i}^{N_{mol}} \mathbf{d}_i \tag{5.63}$$

where the index i now counts the N_{mol} molecules in volume V. The prefactor $g(\omega)$ is a smooth function of frequency, modulating the intensity of the absorption bands but not their position (compare the atomic form factor in Eq. 5.13).

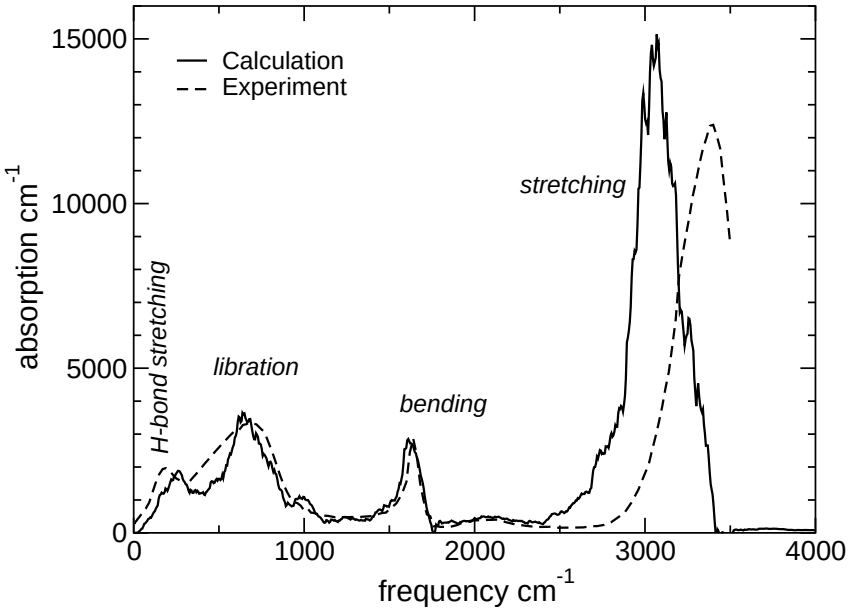


Figure 5.12: Infra red absorption spectrum of liquid water computed from the Fourier transform of the polarization time correlation function compared to experiment (the redshift of the stretching band is an artefact of the computational method used).

The result of such a calculation for the infra red absorption of liquid water is shown in Fig. 5.12. The polarization as well the interatomic forces driving the dynamics in this simulation were obtained using electronic structure calculation methods rather than from a force field, allowing for a more accurate determination of the polarization fluctuations. Comparing to the velocity autocorrelation spectrum of Fig. 5.11 we see that the location and width of the bands are fairly correctly predicted by the velocity autocorrelation but that there can be substantial differences in the relative intensities.

5.8 Velocity autocorrelation and diffusion

5.8.1 Diffusion from mean square displacement

Diffusion is one of the characteristic traits of liquids: Particles in liquids wander around without having a fixed equilibrium position. This is mathematically best expressed by the mean square displacement taking the position at $t = 0$ as reference (origin)

$$\Delta R^2(t) = \langle (\mathbf{r}_i(t) - \mathbf{r}_i(0))^2 \rangle \tag{5.64}$$

where the angular brackets again denote an average over an equilibrium ensemble of initial configuration at $t = 0$. In a perfect solid the mean square displacements is bound, it soon reaches its maximum value

$$\Delta R^2(t) \leq R^2_{max} \tag{5.65}$$

R_{max} is determined by the amplitude of the thermally excited vibrations. In liquids, in contrast, after some initial oscillation, the square displacement of particles continues to increase proportional with time.

$$\Delta R^2(t) = \langle (\mathbf{r}_i(t) - \mathbf{r}_i(0))^2 \rangle = 6D_s t \qquad t \gg 0 \tag{5.66}$$

where D_s is the self diffusion constant. In real solids, at high temperatures, there is some residual diffusion. This is the result of vacancies and other defects. The diffusional motion, however, is very slow and usually highly activated (the transport coefficient distinguishing liquids from solids is viscosity, which rigorously vanishes in a crystal).

Dynamical quantities, such as the mean square displacement of Eq. 5.64, are directly accessible in MD simulation. Fig. 5.13 gives an example of such a numerical measurement for a noble gas liquid. Again reduced units are used. So, length in measured in units of repulsive

Figure 5.13: Diffusion in a Lennard Jones liquid monitored by the mean square atomic displacement (Eq. 5.64) against time. Note the difference between the displacement at low temperature ($T^* = 0.85$) close to freezing (compare Eq. 5.65) and high temperature ($T^* = 1.9$). The selfdiffusion constant is estimated from the slope of a linear fit to the mean displacement using the Einstein relation Eq. 5.66.

core diameter σ

$$l^* = \frac{l}{\sigma} \tag{5.67}$$

Temperature is indicated as a fraction of well-depth ϵ

$$T^* = \frac{k_B T}{\epsilon} \tag{5.68}$$

and the unit of time is a combination of energy, length and mass derived from the expression for kinetic energy

$$t^* = t \left(\frac{\epsilon}{m\sigma^2} \right)^{1/2} \tag{5.69}$$

5.8.2 Long time behavior and diffusion

The mean square displacement as defined in Eq. 5.64 can be considered as a time correlation function of position, but a peculair one, since it keeps on increasing in time instead of dying out. Howwever the mean square displacement is also related to the long time behavior of the velocity autocorrelation function (the "tail" in figure 5.5). The basic reason is, ofcourse, that displacement is the integral of velocity

$$\mathbf{r}(t) - \mathbf{r}(0) = \int_0^t dt' \, \mathbf{v}(t') \tag{5.70}$$

the velocity auto correlation and average square displacement $\Delta R^2(t)$ of Eq. 5.64 must be related. To make this relation explicit we evaluate the time derivative

$$\frac{d}{dt} \Delta R^2(t) = \frac{d}{dt} \int_0^t dt' \int_0^t dt'' \langle \mathbf{v}(t') \cdot \mathbf{v}(t'') \rangle = 2 \int_0^t dt' \, \langle \mathbf{v}(t) \cdot \mathbf{v}(t') \rangle \tag{5.71}$$

Using the invariance of the origin of time (Eq. 5.39)

$$\frac{d}{dt} \Delta R^2(t) = 2 \int_0^t dt' \, \langle \mathbf{v}(t - t') \cdot \mathbf{v}(0) \rangle = 2 \int_0^t d\tau \, \langle \mathbf{v}(\tau) \cdot \mathbf{v}(0) \rangle \tag{5.72}$$

where the second identity is obtained by a change of integration variables form $t-t'$ to τ. Then substituting Eq. 5.64 we see that we must have for sufficiently large t.

$$D_s = \frac{1}{3}\int_0^t d\tau \, \langle \mathbf{v}(\tau) \cdot \mathbf{v}(0) \rangle \tag{5.73}$$

Comparing to Eq. 5.48 we see that the selfdiffusion coefficient can be compared to a "relaxation rate" obtained by integration of the exponential tail of the unnormalized velocity autocorrelation. This approach is an alternative to estimation of D_s from mean square displacements. For reasons of numerical accuracy, however, the mean square displacement method is preferred.

5.9 Appendix 3: Code samples

5.9.1 ∗Computation of time correlation functions

Numerical evaluation of the velocity autocorrelation function is more involved than the averaging procedures discussed so far, because the MD estimator for time correlation functions of Eq. 5.30 requires continuous availability of past configurations. Storage of a full trajectory in a very long array and repeated recall of data from this array can be inconvenient and slow, in particular on computers with limited I/O capabilities. However, the maximum time span over which correlations are of interest is often considerably smaller than the full run length. A clever scheme applied in most MD codes is keeping only as many time points in memory as needed relative to the current time step. The data of times further back are continuously overwritten by the new incoming data while the run progresses. The code below is an implementation of this cyclic overwriting scheme. The integer *memor* is the maximum number of time steps over which correlations are collected. Thus, the array *cvv* in which the products of the velocities vx, vy and vz are accumulated is of length *memor*. The first element $cvv(1)$ contains the $t = 0$ correlation, i.e the velocity variance. rvx, rvy and rvz are two dimensional memory arrays of size $natom \times memor$ for storage of past velocities.

```
subroutine time_cor (natom, vx, vy, vz, memor, msampl, ntim, cvv, rvx, rvy, rvz)
ltim := mod (msampl, memor) + 1  \* memory array slot to be filled or replaced *\
                \* add current velocity at end of memory array or replace oldest slot *\
do i := 1, natom
     rvx(i, ltim) := vx(i)
     rvy(i, ltim) := vy(i)
     rvz(i, ltim) := vz(i)
enddo
msampl := msampl + 1  \* update counter for total number of calls to time_cor *\
mtim := min (msampl, memor)                    \* number of bins already filled *\
jtim := ltim                    \* first bin of auto correlation array to be updated *\
do itim := 1, mtim                             \* loop over current and past velocities *\
      \* sum products of current velocity and velocities stored in memory array *\
     cvvt := 0
     do i := 1, natom
          cvvt := cvvt + vx(i) * rvx(i, itim) + vy(i) * rvy(i, itim) + vz(i) * rvz(i.itim)
     enddo
          \* if last update was for current time switch to oldest time in memory *\
```

```
      if (jtim = 0) jtim := memor
      cvv(jtim) := cvv(jtim) + cvvt \* add to correct bin of autocorrelation array *\
      ntim(jtim) := ntim(jtim) + 1          \* and count how often this is done *\
      jtim := jtim - 1
enddo
end
```

Code 5.4: Accumulation of velocity auto correlation using cyclic memory array

The manipulation of the address indices *itim* and *jtim* of the memory respectively auto correlation arrays is rather subtle. *itim* and *jtim* depend on the integral number of calls to *time_cor* which is updated in *msampl*. The procedure *time_cor* can be performed every time step or also with a period *nsampl*. The time interval between two consecutive sampling points is therefore $nsampl \times dt$, where dt is the time step as in Code 4.30. In order to obtain the velocity autocorrelation at the end of the run, the array elements $cvv(i)$ are normalized by a factor corresponding to number of times $ntim(i)$ the bin i has been accessed.

```
mtim := min(msampl, memor)        \* in case that nstep < nsampl * memor *\
dtime := nsampl * dt
fact := ntim(1)/cvv(1)                \* normalization by velocity variance *\
do itim := 1, mtim
      tim(i) := itim * dtime
      cvv(i) := fact * cvv(i)/ntim(i)
enddo
```

Code 5.5: Final normalization of velocity auto correlation

Having determined the velocity autocorrelation we can evaluate its integral and use Eq. 5.73 to compute the selfdiffusion coefficient. This is in practice not a very accurate way of estimating self diffusion constants. The problem is that the integral is dominated by contributions from the tail of the velocity auto correlation at long times. This is exactly the part of the data where statistics tends to be poor (see Eq. 5.30).

An alternative method which usually gives better results for runs of moderate length is determining D_s from the slope of the mean square displacement using Eq. 5.64. The invariance under translation of the reference time (Eq 5.39) also applies to the mean square displacement $\Delta R^2(\tau)$, and this correlation function can be estimated in MD using a trajectory sum similar to Eq. 5.30. Replacing the products of velocities, by differences of positions Code 5.4 can be easily modified for the sampling of $\Delta R^2(\tau)$. It is important to continue the simulation for a sufficiently long time to be able to verify that in the long time limit $\Delta R^2(\tau)$ is linear with τ.

Project E: *Determine the velocity autocorrelation function for the liquid argon system of Project C. Compute also the self diffusion coefficient from the mean square displacement. How long a trajectory is needed for reasonable convergence of the self diffusion coefficient?*

Chapter 6

Controlling dynamics

6.1 Constant temperature molecular dynamics

The MD algorithms presented in Chapter 4 are numerical procedures for solving Newton's equation of motion and, therefore, generate the micro canonical (NVE) ensemble (Eq.4.38). Most experimental systems of interest are in thermal (T) and mechanical (P) equilibrium with their environment. Therefore, for the purpose of comparing to experiment it would be better if simulations could be performed under similar thermodynamic conditions. In this section we will look at an algorithm for sampling the canonical ensemble using modified Newtonian equations of motion. This is the famous Nosé constant temperature MD method[2, 3, 4, 5]. Similar algorithms have been developed for constant pressure MD which will not be discussed here (see AT, and references [6, 5]).

6.1.1 Nosé dynamics

The basic idea of Nosé's approach[2, 3] to constant temperature MD is extending Newton's equations of motion with a special friction term which forces the dynamics to sample the isothermal instead of the microcanonical ensemble which it would (ideally) reproduce without these additional interactions. The modified Newtonian equations of motion for a system coupled to a Nosé–Hoover thermostat are

$$\begin{aligned} \ddot{\mathbf{r}}_i &= \frac{\mathbf{f}_i}{m_i} - \zeta\,\dot{\mathbf{r}}_i \\ \dot{\zeta} &= \frac{1}{Q}\left[\sum_i^N m_i \dot{\mathbf{r}}_i^2 - 3Nk_BT\right] \quad . \end{aligned} \tag{6.1}$$

The friction coefficient ζ fluctuates in time and responds to the imbalance between instantaneous kinetic energy (first term in the force for ζ) and the intended canonical average (second term, see also Eq. 4.48). Q is a fictitious mass that controls the response time of the thermostat. As a result of these frictional forces the Nosé–Hoover dynamics is not a regular mechanical system because there is energy dissipation. Indeed the time derivative of energy of the system

is now finite. of times before

$$\begin{aligned}\frac{d\mathcal{H}}{dt} &= \sum_{j=1}^{N}\left[\dot{\mathbf{r}}_j\cdot\frac{\partial\mathcal{H}}{\partial\mathbf{r}_j}+\dot{\mathbf{p}}_j\cdot\frac{\partial\mathcal{H}}{\partial\mathbf{p}_j}\right]\\ &= \sum_{i}^{N}\left[\dot{\mathbf{r}}_i\cdot\frac{\partial\mathcal{V}}{\partial\mathbf{r}_i}+\mathbf{p}_i\cdot\frac{\dot{\mathbf{p}}_i}{m_i}\right]=\sum_{i}^{N}\left[-\dot{\mathbf{r}}_i\cdot\mathbf{f}_i+\mathbf{p}_i\cdot\frac{\dot{\mathbf{p}}_i}{m_i}\right]\end{aligned}$$

but now we have to substitute the modified equation of motion Eq. 6.1

$$\frac{d\mathcal{H}}{dt}=\sum_{i}^{N}\left[-\dot{\mathbf{r}}_i\cdot\mathbf{f}_i+\mathbf{p}_i\cdot\left(\frac{\mathbf{f}_i}{m_i}-\zeta\,\dot{\mathbf{r}}_i\right)\right]=-\,\zeta\sum_{i}^{N}m_i\dot{\mathbf{r}}_i^2 \tag{6.2}$$

and we are left with a finite energy derivative proportial to the friction coefficinet and the kinetic energy $\mathcal{K}$

$$\frac{d\mathcal{H}}{dt}=-2\zeta\,\mathcal{K} \tag{6.3}$$

However, the dissipation by the friction term in Eq. 6.1 is rather peculiar: it can have positive but also negative sign, and there is, in fact, an energy quantity $\tilde{\mathcal{H}}$ that is conserved by the Nosé dynamics. $\tilde{\mathcal{H}}$ is the sum of the energy $\mathcal{H}$ of the particle system (Eq. 4.14), the kinetic energy associated with the dynamical friction and a term involving the time integral of the friction coefficient.

$$\tilde{\mathcal{H}}=\mathcal{H}+\frac{Q}{2}\zeta^2+3Nk_BT\int_0^t dt'\zeta(t')\quad . \tag{6.4}$$

Inserting in total time derivative of this "extended" hamiltonian $\tilde{\mathcal{H}}$

$$\frac{d\tilde{\mathcal{H}}}{dt}=\frac{d\mathcal{H}}{dt}+Q\zeta\dot{\zeta}+3Nk_BT\zeta$$

the equation of motion for the dynamical friction constant (Eq. 6.1)

$$\frac{d\tilde{\mathcal{H}}}{dt}=\frac{d\mathcal{H}}{dt}+\zeta\left[\sum_{i}^{N}m_i\dot{\mathbf{r}}_i^2-3Nk_BT\right]+3Nk_BT\zeta=\frac{d\mathcal{H}}{dt}+\zeta\sum_{i}^{N}m_i\dot{\mathbf{r}}_i^2$$

Comparing to Eq. 6.3 we see that the change in energy of the system is exactly canceled by the thermostat and thus

$$\frac{d\tilde{\mathcal{H}}}{dt}=0 \tag{6.5}$$

The Nosé dynamics of course has been "designed" in this way.

6.1.2 How Nosé-thermostats work

The relations 6.4 and 6.5 can be used for a qualitative explanation of the functioning of the Nosé thermostat. Suppose the system is during the time interval t_1, t_2 in a stationary state. Under such equilibrium conditions we can neglect the difference in kinetic energy of the thermostat at time t_1 and t_2 since

$$\frac{Q}{2}\zeta(t_1)^2\approx\frac{Q}{2}\zeta(t_2)^2\approx\frac{k_BT}{2} \tag{6.6}$$

Inserting in Eq. 6.4 gives

$$\mathcal{H}(t_2) - \mathcal{H}(t_1) + 3Nk_BT \int_{t_1}^{t_2} dt\zeta \approx \tilde{\mathcal{H}}(t_2) - \tilde{\mathcal{H}}(t_1) = 0 \quad . \tag{6.7}$$

The change in the total atomic energy $\Delta\mathcal{H} = \mathcal{H}(t_2) - \mathcal{H}(t_1)$ is correlated to the time integral over the friction coefficient, i.e. the energy dissipated by the thermostat. Depending on the sign of ζ the thermostat has either a cooling or heating effect on the atoms.

$$\begin{aligned} \zeta \;\; &> \;\; 0 \rightarrow \Delta\mathcal{H} \approx -3Nk_BT \int_{t_1}^{t_2} dt\zeta < 0 \;\; \text{cooling} \\ \zeta \;\; &< \;\; 0 \rightarrow \Delta\mathcal{H} \approx -3Nk_BT \int_{t_1}^{t_2} dt\zeta > 0 \;\; \text{heating} \quad . \end{aligned} \tag{6.8}$$

The astonishing property of the Nosé thermostat is that, provided the extended dynamics of Eq. 6.1 is ergodic, it is also canonical in the thermodynamic sense, i.e. the states along a trajectory of the atomic system are distributed according to the isothermal ensemble Eq. 4.39. The proof, given by Nóse is rigorous and will not be repeated here. It can be found in Ref. [2] (see also FS). How the peculiar dynamics invented by Nosé accomplishes this feat can be understood in a more intuitive way from an enlightning argument by Hoover[3], which come close to a heuristic proof.

6.1.3 ∗Technical implementation of Nosé scheme

The forces in the Newtonian equations of motion Eq. 6.1 for a system of N atoms coupled to a Nosé–Hoover thermostat depend on velocities. A natural choice for a numerical integration scheme for these dynamical equations is therefore the velocity Verlet algorithm introduced in section 4.1.3. Our dynamical variables are the particle positions $\mathbf{r}_i$ and η the time integral of the friction coefficient ζ. The velocity Verlet integrators Eq. 4.25 for position and 4.29 for velocity give for these variables

$$\begin{aligned} \mathbf{r}_i(\delta t) &= \mathbf{r}_i(0) + \dot{\mathbf{r}}_i(0)\delta t + \left[\frac{\mathbf{f}_i(0)}{m_i} - \dot{\eta}(0)\dot{\mathbf{r}}_i(0)\right]\frac{\delta t^2}{2} \\ \eta(\delta t) &= \eta(0) + \dot{\eta}(0)\delta t + f_\eta(0)\frac{\delta t^2}{2Q} \\ \dot{\mathbf{r}}_i(\delta t) &= \dot{\mathbf{r}}_i(0) + \left[\frac{\mathbf{f}_i(0)}{m_i} - \dot{\eta}(0)\dot{\mathbf{r}}_i(0) + \frac{\mathbf{f}_i(\delta t)}{m_i} - \dot{\eta}(\delta t)\dot{\mathbf{r}}_i(\delta t)\right]\frac{\delta t}{2} \\ \dot{\eta}(\delta t) &= \dot{\eta}(0) + [f_\eta(0) + f_\eta(\delta t)]\frac{\delta t}{2Q} \end{aligned} \tag{6.9}$$

where we have simplified our notation by setting the current time $t = 0$. f_η is the force on the thermostat

$$f_\eta = \sum_{i=1}^{N} m_i\dot{\mathbf{r}}_i^2 - 3NkT \tag{6.10}$$

The coordinate update requires as input the velocities at the advanced time δt. These data are needed for the computation of the friction forces. However, these velocities only become available after the update of position. Eq. 6.9 are, therefore, a selfconsistent set of equations

which have to be solved by iteration. The velocities $\dot{\mathbf{r}}_i^{(k)}$ and $\dot{\eta}_{(k)}$ at step k are obtained from the values at the previous iteration step by

$$\dot{\mathbf{r}}_i^{(k)}(\delta t) = \frac{\dot{\mathbf{r}}_i(0) + \left[\frac{\mathbf{f}_i(0)}{m_i} - \dot{\eta}(0)\dot{\mathbf{r}}_i(0) + \frac{\mathbf{f}_i(\delta t)}{m_i}\right]\frac{\delta t}{2}}{1 + \frac{\delta t}{2}\dot{\eta}^{(k-1)}(\delta t)}$$
$$\dot{\eta}^{(k)}(\delta t) = \dot{\eta}(0) + \left[f_\eta(0) + f_\eta^{(k)}(\delta t)\right]\frac{\delta t}{2Q} \tag{6.11}$$

A good initial guess for $\dot{\eta}$ is

$$\dot{\eta}^{(0)}(\delta t) = \dot{\eta}(-\delta t) + 2f_\eta(0)\frac{\delta t}{Q} \tag{6.12}$$

With the introduction of iteration, rigorous time reversibility which was one of the key features of the Verlet algorithm no longer holds. With the help of the same advanced techniques based on Liouville operators[1] it is possible to derive an explicitly time reversible integrator for Nosé–Hoover dynamics[1, 5].

6.2 Constrained dynamics

In this section, yet another tool typical for the MD approach is presented, namely a scheme for fixing geometric functions of particle coordinates, such as distance or bond-angle, by the application of mechanical (holonomic) constraints. This method was originally developed to deal with the large gap in time scales between intra molecular and intermolecular dynamics. This so-called method of constraints is, however, more general and is, for example, used in thermodynamic integration schemes for the computation of (relative) free energy

6.2.1 Multiple time scales

To appreciate this problem of fast versus slow time scales a very brief introduction to the modelling of the forces stabilizing molecular geometry is helpful. In first approximation these so called bonding forces can be described by harmonic potentials. Thus, if atom 1 and 2 in a molecule are connected by a chemical bond with an equilibrium length of d_0, the simple oscillator potential

$$v\left(r_{12}\right) = \frac{1}{2}k_s\left(r_{12} - d_0\right)^2 \tag{6.13}$$

is sufficient to impose an (average) interatomic distance of r_0. The spring constant k_s can be adjusted to reproduce the frequency of the bond stretch vibration. Similarly if a third atom 3 is bonded to atom 2, the bond angle is held in place by the potential

$$v\left(\theta_{123}\right) = \frac{1}{2}k_b\left(\theta_{123} - \theta_0\right)^2 \tag{6.14}$$

where θ_{123} is the angle between the interatomic vectors $\mathbf{r}_{12} = \mathbf{r}_1 - \mathbf{r}_2$ and $\mathbf{r}_{32} = \mathbf{r}_3 - \mathbf{r}_2$ with equilibrium value θ_0. The spring constant for bond bending is k_b.

These spring constants are stiff and intra molecular forces can be are several orders of magnitude stronger than inter-molecular forces. For example, typical vibration frequencies of bonds between carbon, nitrogen or oxygen atoms are in the range of 1000 cm^{-1} or more. Bonds of hydrogen atoms to these first row atoms can have frequencies over 3000 cm^{-1}. In contrast,

The dynamics of interest in molecular liquids proceeds on a picosecond time scale, i.e in the 10 to 100 cm^{-1} range. The time step in the Verlet algorithm scales with the inverse of the maximum frequency in the system. This forces us to use a time step much shorter than needed for the integration of the equations of motion driven by intermolecular interactions, which in the study of liquids is our main topic of interest.

The bulk of the cpu time required for completing a full MD iteration is spent on the determination of the non-bonded intermolecular forces. Computation of intramolecular forces is comparatively cheap. This suggests that the efficiency of MD algorithms could be improved significantly if the time interval separating non-bonded force calculations could be stretched to times comparable to the corresponding intermolecular time step. This amounts to MD iteration with a different time step for the inter- and intermolecular forces. This is the idea behind multiple time step algorithms. The idea seems simple, but the search for stable multiple time step algorithms has a history full of frustrations. It was discovered that it is far from obvious how to insert small time steps which iterate a subset of the forces without introducing inconsistencies in the particular propagator scheme that is employed. Mismatches invariably lead to enhanced energy drift. Only recently a satisfactory solution to this consistency problem was found by Tuckerman, Martyna and Berne [1, 5]. They showed that their Trotter factorization method can also be used to generate stable two stage discrete time propagators, for separate updates of intramolecular and intermolecular dynamics. Again, even though a highlight of modern MD, this technique will have to be skipped in this short course.

6.2.2 Geometric constraints

A drastic solution to the problem of disparate time scales is to ignore intra molecular dynamics all together and keep the geometry of the molecules fixed. This was, until the development of stable and reliable multi time step algorithms, the approach used in the majority of MD codes for molecular systems. It is still a very useful and efficient method for simple molecules (water, methanol, ammonia etc) for which completely rigid models are a good first approximation. More complex molecules, in particular chain molecules with torsional degrees of freedom, require flexible (or partly flexible) force fields. For these systems multi-time step algorithms have definite advantages. Constraint methods are also useful in the computation of free energies Here we give a short introduction of the method of constraints with particularly this application in mind. Geometrical constraints can be expressed as a set of equations for the Cartesian position vectors. For the elementary example of a homo-nuclear diatomic, e.g the N_2 molecule, the only intra molecular degree of freedom is bond length. A bondlength constraint can be either in the form of a constraint on the distance r_{12} but also the quadratic relation (cf. Eq. 6.13)

$$\mathbf{r}_{12} \cdot \mathbf{r}_{12} - d_0^2 = 0 \tag{6.15}$$

will do. A fixed bond angle θ_0 , e.g. the HOH angle in the water molecule, involves three position vectors and can be described by the constraint

$$\frac{\mathbf{r}_{12} \cdot \mathbf{r}_{32}}{|\mathbf{r}_{12}|\,|\mathbf{r}_{32}|} - \cos\theta_0 = 0 \tag{6.16}$$

In general we have M of these constraint relations, specified by M coordinate functions σ_α

$$\sigma_\alpha(\mathbf{r}^N) = 0, \quad \alpha = 1, ...M \tag{6.17}$$

In order to make the dynamics satisfy these constraints the gradients of the σ_α are treated as forces and added to the Newtonian equations of motion

$$\begin{aligned} \mathbf{g}_i &= \sum_\alpha \lambda_\alpha \nabla_i \sigma_\alpha \\ m_i \ddot{\mathbf{r}}_i &= \mathbf{f}_i + \mathbf{g}_i \end{aligned} \tag{6.18}$$

The parameters λ_α define the strength of the constraint force. Their value fluctuates with time and is such that at any instant the constraint forces $\mathbf{g}_i$ exactly cancel the components of the potential forces $\mathbf{f}_i$ which would lead to violation of the constraints. These forces are normal to the hyper-surfaces in coordinate space described by the relations Eq. 6.17. The constraint forces can be obtained by substituting the equation of motion in the second order time derivative of the equations Eq. 6.17 and solving for λ_α. For a rigorous justification of this approach we need to adopt again a more formal treatment of mechanics, namely the method of Lagrange. From the abstract point of view of Lagrange the Newtonian equations of motion are solutions of a variational problem in space-time. This variational problem can be subjected to constraints and the coefficients λ can then be identified with undetermined Lagrange multipliers.

As an illustration let us go through the example of the quadratic bond constraint Eq. 6.15. The equations of motion are

$$m_1 \ddot{\mathbf{r}}_1 = \mathbf{f}_1 + 2\lambda \mathbf{r}_{12}, \quad m_2 \ddot{\mathbf{r}}_2 = \mathbf{f}_2 - 2\lambda \mathbf{r}_{12} \tag{6.19}$$

Differentiating Eq. 6.15 twice with respect to time

$$\frac{d^2\sigma}{dt^2} = 2\ddot{\mathbf{r}}_{12} \cdot \mathbf{r}_{12} + 2\dot{\mathbf{r}}_{12} \cdot \dot{\mathbf{r}}_{12} = 0 \tag{6.20}$$

and inserting Eq. 6.19 yields

$$\left[\frac{\mathbf{f}_1}{m_1} - \frac{\mathbf{f}_2}{m_2}\right] \cdot \mathbf{r}_{12} + 2\lambda \left[\frac{1}{m_1} + \frac{1}{m_2}\right] \mathbf{r}_{12}^2 + \dot{\mathbf{r}}_{12}^2 = 0 \tag{6.21}$$

Solving for λ we obtain

$$\lambda = -\frac{\mu}{2r_{12}^2}\left[\left(\frac{\mathbf{f}_1}{m_1} - \frac{\mathbf{f}_2}{m_2}\right) \cdot \mathbf{r}_{12} + \dot{\mathbf{r}}_{12}^2\right] \tag{6.22}$$

where μ is the reduced mass. Substituting in Eq. 6.19 we find for atom 1 the equation

$$m_1 \ddot{\mathbf{r}}_1 = \mathbf{f}_1 - \frac{\mu}{r_{12}^2}\left[\left(\frac{\mathbf{f}_1}{m_1} - \frac{\mathbf{f}_2}{m_2}\right) \cdot \mathbf{r}_{12} + \dot{\mathbf{r}}_{12}^2\right] \mathbf{r}_{12} \tag{6.23}$$

coupled to a similar equation for particle 2.

In Eq. 6.23 we encounter once more velocity dependent forces. Similar to the way we treated the Nosé-Hoover thermostat in section 6.1.3, we can try to find an iterative velocity Verlet scheme to integrate Eq. 6.23. However, in contrast to a thermostat friction force, constraint forces can substantial. Moreover, in hydrogen bonded systems they tend to be highly anisotropic, continuously pulling in the same direction. This leads to rapid accumulation of errors and eventually divergence.

6.2.3 Method of constraints

As shown by Ciccotti, Ryckaert and Berendsen[7, 8], in the case of constraints it is possible to avoid velocity iteration altogether. Their idea was to rigorously satisfy the constraints at the level of the discrete propagator itself. The implementation of this method for the Verlet algorithm has proven to be both very effective and stable. Consider the prediction of Eq. 4.23 for the advanced positions based on potential forces only

$$\mathbf{r}_i^{\mathrm{u}}(t+\delta t) = 2\mathbf{r}_i(t) - \mathbf{r}_i(t-\delta t) + \frac{\delta t^2}{m_i}\mathbf{f}_i(t) \tag{6.24}$$

The suffix u is to indicate that these coordinates will in general violate the constraints. Next we add the constraint forces with the unknown Lagrange multipliers

$$\mathbf{r}_i^{\mathrm{c}}(t+\delta t) = \mathbf{r}_i^{\mathrm{u}}(t+\delta t) + \frac{\delta t^2}{m_i}\sum_\alpha \lambda_\alpha \nabla_i \sigma_\alpha(t) \tag{6.25}$$

and substitute these "corrected" positions in the constraint equations.

$$\sigma_\alpha\left(\{\mathbf{r}_i^{\mathrm{c}}(t+\delta t)\}\right) = \sigma_\alpha\left(\left\{\mathbf{r}_i^{\mathrm{u}}(t+\delta t) + \frac{\delta t^2}{m_i}\sum_\beta \lambda_\beta \nabla_i \sigma_\beta(t)\right\}\right) = 0 \tag{6.26}$$

The result is a set equations for λ_α which can be solved numerically. Again the example of the quadratic bond constraint Eq. 6.15 is very instructive because it can be treated analytically. For this case Eq. 6.26 is quadratic

$$\left[\mathbf{r}_{12}^{\mathrm{c}}(t+\delta t)\right]^2 - d_0^2 = \left[\mathbf{r}_{12}^{\mathrm{u}}(t+\delta t) + \frac{2\delta t^2}{\mu}\lambda \mathbf{r}_{12}(t)\right]^2 - d_0^2 = 0 \tag{6.27}$$

The root with the correct $\delta t \to 0$ limit is (assuming that $\mathbf{r}_{12}^2(t) = d_0^2$)

$$\lambda = \frac{-\mathbf{r}_{12}(t)\cdot\mathbf{r}_{12}^{\mathrm{u}}(t+\delta t) + \sqrt{\left[\mathbf{r}_{12}(t)\cdot\mathbf{r}_{12}^{\mathrm{u}}(t+\delta t)\right]^2 - d_0^2\left[\mathbf{r}_{12}^{\mathrm{u}}(t+\delta t)^2 - d_0^2\right]}}{2d_0^2\,\delta t^2\mu^{-1}} \tag{6.28}$$

By satisfying the constraint exactly, numerical errors have been eliminated in this approach at virtually no additional computational cost. A similar constraint algorithm has been developed for velocity Verlet[9].

Bibliography

[1] M. Tuckerman, G. J. Martyna, and B. J. Berne, J. Chem. Phys. **97** 1990, (1992).

[2] S. Nosé, J. Chem. Phys. **81**, 511 (1984).

[3] W.G. Hoover, Phys. Rev. A**31**, 1695 (1985).

[4] G. Martyna, M. L. Klein, and M. Tuckerman, J. Chem. Phys. **97**, 2635 (1992).

[5] G. Martyna, M. Tuckerman, D. J. Tobias, and M. L. Klein, Mol. Phys. **87**, 1177 (1996).

[6] H. C. Andersen, J. Chem. Phys. **72**, 2384 (1980).

[7] J. P. Ryckaert, G. Ciccotti, and H. J. Berendsen, J. Comp. Phys. **23**, 327 (1977).

[8] J. P. Ryckaert and G. Ciccotti, Comp. Phys. Rep. **4**, 345 (1986).

[9] H. C. Andersen, J. Comp. Phys. **52**, 24 (1983).

[10] C. H. Bennet, J. Comp. Phys. **22**, 245 (1976)

[11] D. Chandler, J. Chem. Phys. **68**, 2951 (1978).

[12] G. M. Torrie, and J. P. Valleau, J. Comp. Phys. **23**, 187 (1977).

[13] E. Carter, G. Ciccotti, J. Hynes, and R. Kapral, Chem. Phys. Lett. **156**, 472 (1989).

[14] M. Sprik, and G. Ciccotti, J. Chem. Phys. **109**, 7737 (1998).

[15] P. G. Bolhuis, C. Dellago, D. J. Chandler, Faraday Discuss. **110**, 42 (1998)